Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya

OPEN DATA SOURCES FOR URBAN MOBILITY

Autor: **Gerard Alonso Bohigas**

Director: **Jorge García Vidal**

Codirector: **Jose Maria Barceló Ordinas**

FIB – Grau en Enginyeria Informàtica

Treball de final de grau

Quadrimestre primavera 2015/2016

Agraïments

M’agradaria agrair primer de tot al grup d’investigació Compnet que m’ha donat l’oportunitat de poder realitzar el projecte amb ells, en especial a Jorge Garcia i a Jose M. Barceló, que m’han orientat a l’hora de prendre decisions.

També m’agradaria agrair als meus companys de grup Albert , Biel i Santi amb els quals he treballat conjuntament en algunes parts de diferents projectes.

Resum

L'objectiu principal d'aquest projecte és automatitzar l'extracció de dades del serveis de bicicletes de diferents ciutats, i integrar-ho dintre de la plataforma actual Commsensum que conté dades de les estacions que prenen mida de la qualitat de l'aire. Finalment, mostrar en una aplicació web la combinació de les dues dades en un mapa.

Resumen

El objetivo principal de este proyecto es automatizar la extracción de datos de Servicios de bicicletas de diferentes ciudades, e integrar-lo dentro de la plataforma actual Commsensum que contiene datos de las estaciones que miden la calidad del aire. Finalmente, mostrar en una aplicación web la combinación de los dos datos en un mapa.

Abstract

The main aim of this project is to automatize the data extraction process from public bicycles services from different cities, and integrate all the data inside the current platform Commsensum that contains air quality data. Finally, show both datasets in a web page in a map-fashion.

Índex

1. Introducció

1.1. Formulació del problema

Cada cop hi ha més informació **opendata**[1] publicada arreu del món, però aquesta està en formats diversos, de manera que dificulta la seva utilització pràctica en aplicacions. Fins i tot una ciutat com Barcelona, model en quant a **Smartcity**[2], està afectada per aquest problema.

El concepte de compartir informació de manera pública per la seva utilització per la millora de la ciutat és molt bona idea. Això és només el primer pas, publicar la major quantitat d'informació interessant possible. Però, l'objectiu principal és treure-li més partit agrupant tota aquesta informació relacionada per tal de que tingui més valor i sigui més fàcil utilitzar-la. Per exemple, si agrupem la informació obtinguda de la pol·lució de varis sensors localitzats per tota Barcelona mostrant-la de manera entenedora en un mapa, aquesta tindrà molt més valor per qui la consulti.

Podem anar més enllà i combinar aquesta informació amb la obtinguda pel servei de Bicing a Barcelona, de manera que es podria considerar la idea de circular en bicicleta per les zones menys contaminades. Aquest és només un exemple, ja que les possibilitats poden ser innumerables, tenint en compte tota la informació que es pot obtenir en cada ciutat.

Un altre aspecte és el fet de poder emmagatzemar certa informació en un format estàndard en una aplicació intermèdia, entre les fonts d'informació de dades obertes i les aplicacions. Aquesta aplicació guardarà la informació en un mateix format de manera que les aplicacions no s'hauran de preocupar del problema de tractar informació en diferents formats. I no només això, sinó que una aplicació amb un cert objectiu en una ciutat, podrà utilitzar-se en una altre ciutat pràcticament sense ser modificada, ja que aquesta obtindrà les dades de la aplicació intermèdia. L'únic procés feixuc que s'hauria de realitzar és l'extracció i emmagatzematge de la informació de diferents fonts de dades obertes, que segurament tindran un format distint.

Resumint, el que es pretén és accelerar la segona fase, que amb el pas del temps es farà més necessària en les ciutats més modernes. Cada cop es generarà més informació, i tenint en compte el gran creixement de la utilització del **Internet of Things**[3], és una evolució lògica i natural. Així doncs cal recalcar que aquesta informació segurament no estarà estructurada, i per empitjorar-ho encara més, estarà en diferent formats.

[1] Opendata: són tots aquells conjunts de dades que es posen a disposició del públic i poden ser reutilitzats i tornats a publicar sense cap restricció.
[2] Smartcity: són les que estan dotades de mecanismes basats en les tecnologies de la societat de la informació i la comunicació enfocats a millorar tant la gestió dels diferents serveis com la qualitat de vida dels seus habitants.
[3] Internet of things: es refereix, en termes d'informàtica, a una xarxa d'objectes de la vida quotidiana interconnectats

La possibilitat de posar la teva gota d'aigua per poder ajudar a millorar la qualitat de vida de les persones facilitant la informació de la pol·lució de les ciutats és un factor més que he tingut en compte a l'hora de triar el projecte. Facilitar el flux d'informació publica que es genera en una Smartcity és molt important.

1.2. Entorn de desenvolupament

El desenvolupament d'un projecte de final de grau consisteix en que l'estudiant posi en pràctica totes les habilitats i coneixements adquirits al llarg del Grau. Però no és només això, sinó que el fet de poder-lo portar a terme dintre d'un grup d'investigació de la pròpia universitat el converteix en més atractiu segons el meu punt de vista, ja que el impacte que podria arribar a tenir el projecte podria augmentar gràcies a aquest fet. Els temes en els que investiga aquest grup, com el internet de les coses o les ciutats intel·ligents són àrees força interessants per mi, i tenir l'oportunitat de poder treballar amb professors experts en la matèria és una motivació extra.

Un altre factor que em va motivar força va ser la oportunitat de poder contribuir a millorar el medi ambient i la qualitat de vida de les persones. Sempre m'ha importat el medi ambient, i el fet de poder fer més fàcil la vida a les persones. Inicialment, quan encara no estava del tot definit el projecte, no estava gaire clar si seria possible o no. Però, quan es va poder definir en més detall, va ser possible tenir-ho en compte.

El projecte s'ha realitzat a la universitat al grup d'investigació Computer Networks and Distributed Systems (CNDS) , concretament al subgroup Compnet, que pertany al departament d'arquitectura de computadors, i que el formen principalment Jorge Garcia Vidal, Jose M. Barceló Ordinas i Manel Guerrero-Zapata. Aquest grup treballa a part dels temes anomenats anteriorment, en altres com xarxes sense fils. Se m'ha facilitat una aula i un ordinador per poder treballar més còmodament a dintre de la pròpia universitat. A la vegada, s'ha compartit la aula amb altres estudiants que o bé estaven realitzant el projecte final de grau com jo, o un màster, persones amb les quals s'ha pogut intercanviar idees i treballar en grup en algunes parts del projecte.

1.3. Objectius

El projecte està clarament dividit en diferents objectius, tant tècnics com generals. Primer de tot es descriuen els objectius generals i posteriorment els tècnics.

1.3.1. Objectius generals

El primer objectiu és conscienciar a la població de quina és el nivell de contaminació de l'aire en diverses àrees de la seva ciutat indirectament gràcies a la informació que és mostrarà

conjuntament amb la de les estacions de bicicletes disponibles. Així els usuaris sabran quines zones estan pitjor i podran triar rutes alternatives.

També es pretén impulsar la utilització de vehicles de transports sostenibles, i així aconseguir una ciutat més neta.

Accelerar la utilització de dades obertes públiques a nivell internacional mostrant el seu possible ús. També mostrar la importància dels estàndards en quan a la utilització de la web semàntica amb **RDF**[4] i **OWL**[5].

1.3.2. Objectius tècnics

Desenvolupar un **crawler**[6] recol·lector de dades de mobilitat de dos fonts diferents, en concret dades de mobilitat en bicicleta de fonts d'informació d'opendata. La primera font serà de Barcelona[2], és a dir, Bicing. La segona font serà d'una altre ciutat d'Espanya com per exemple Saragossa. Podem dividir aquest punt en dos processos diferents que s'executaran periòdicament, ja que les fonts probablement estaran en dos estructures de dades diferents.

- ***Crawler Barcelona***: Obtenir informació de opendata.bcn.cat, parsejar-la i enviar-la a la plataforma.
- ***Crawler Zaragoza***: Obtenir informació de zaragoza.es/api, parsejar-la i enviar-la a la plataforma.

Definir o adaptar una ontologia de coneixement amb el llenguatge OWL on es pugui representar tota la informació que s'obtindrà pel que fa el servei de bicicletes públic de cada ciutat. Caldrà crear-la, o bé des de zero, o partint d'alguna existent publicada en algun altre projecte o a internet per algun col·laborador.

Emmagatzemar la informació de manera ordenada a la **BBDD**[7] de la plataforma. S'haurà de decidir la millor estructura conjuntament amb els altres dos projectes directament relacionats que també es realitzen a la vegada per dos altres estudiants.

Distribuir la informació públicament a través del format RDF amb una **API REST**[8]. S'haurà de decidir la millor manera de distribuir-la conjuntament amb els altres dos projectes directament relacionats.

[4] RDF (Resource Description Framework): és un model de dades per definir recursos i les seves relacions.
[5] OWL (Web Ontology Language): és un llenguatge per definir ontologies mitjançant la descripció detallada de propietats i classes.
[6] Crawler: és un programa que sistemàticament navega a la WWW per crear un índex de dades.
[7] BBDD (Base de dades): és un conjunt de estructurat de dades.
[8] API REST (Application Programming Interface Representational State Transfer) : és una tecnologia que permet distribuir dades a través de HTTP retornant fitxer en format estàndard com RDF/XML o RDF/JSON.

Crear una aplicació que mostri la informació tant de l'estat de les bicicletes com de la pol·lució a la ciutat. Es mostrarà tot el contingut en un mapa utilitzant la API de google maps. Aquesta aplicació obtindrà la informació que estarà distribuïda públicament en RDF. Es partirà d'una aplicació que mostra l'estat dels diferents sensors en un mapa, desenvolupada per una altre estudiant de màster.

2. Gestió del projecte

2.1. Estat de l'art

En aquesta secció es descriuen el context del projecte breument, els actors implicats, les solucions existents i les tecnologies seleccionades.

2.1.1. Context

La introducció ja fa una explicació contextualitzada, així doncs, en aquesta secció es fa un resum dels punts més importants.

En l'actualitat Barcelona és una de les ciutats punteres en quant a Smartcity i té força informació opendata, però pateix el problema de la fragmentació de la informació en diferents formats. Així doncs les aplicacions que utilitzen aquesta informació generada per la ciutat depenen d'aquests formats.

El que es pretén per millorar-la és la utilització d'una aplicació intermèdia que gestioni de manera eficaç aquesta informació. El seu objectiu serà obtenir informació pública, l'emmagatzemarà de manera estàndard segons una ontologia que es definirà, i la distribuirà en format estàndard RDF.

Creant aquest procés és guanyarà en portabilitat, perquè una aplicació que s'alimenta de dades de l'aplicació intermèdia, podrà funcionar sense fer-li cap canvi a diferents ciutats. El que s'haurà de canviar serà el procés d'obtenció i emmagatzemat de les dades a l'aplicació intermèdia.

També és realitzarà una aplicació que obtindrà les dades existents dels sensors que fan mesures de la qualitat de l'aire i de les dades de l'estat de les estacions de bicicletes de la ciutat desitjada. D'aquesta manera es veurà la utilitat del procés intermedi a la pràctica, creant aquesta aplicació web que mostrarà la fusió de les dades.

2.1.2. Actors implicats

El projecte va dirigit a qualsevol persona interessada en el medi ambient, que utilitzi o no el servei de bicicletes de la ciutat. També va dirigit a tots els futurs usuaris de la plataforma, que poden estar interessats en aplicar aquesta estratègia a les seves aplicacions a les seves ciutats o països.

La aplicació final serà utilitzada per persones interessades en saber la qualitat de l'aire i la informació de les estacions de bicicletes a la seva ciutat, podent-la consultar a la vegada amb aquesta.

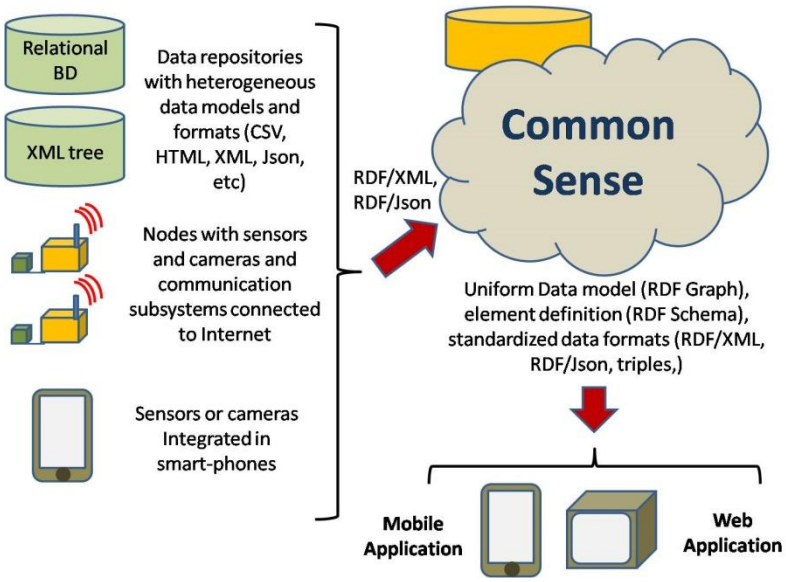
Els beneficiats dels resultats seran moltes persones, tant els usuaris de la aplicació, com els que apliquin aquest projecte a la seva ciutat, millorant la qualitat de vida dels ciutadans i conscienciar-los en el medi ambient. També els membres del grup d'investigació el qual va crear aquest projecte, podent usar-lo, millorar-lo i crear-ne d'altres derivats.

2.1.3. Solucions existents

Aquest projecte és la millora de varis projectes ja fets en anterioritat que a continuació es detallaran. A la vegada, hi ha dos altres projectes directament relacionats amb aquest, que possiblement tindran una part comuna. Un d'ells el seu objectiu és millorar l'aplicació ja existent que mostra la informació de la qualitat d'aire en un mapa. L'altre el seu objectiu principal és ampliar la cerca d'informació per tots els sensors arreu d'Espanya i d'alguna ciutat de Àustria i Itàlia, i poder guardar-la a la plataforma existent.

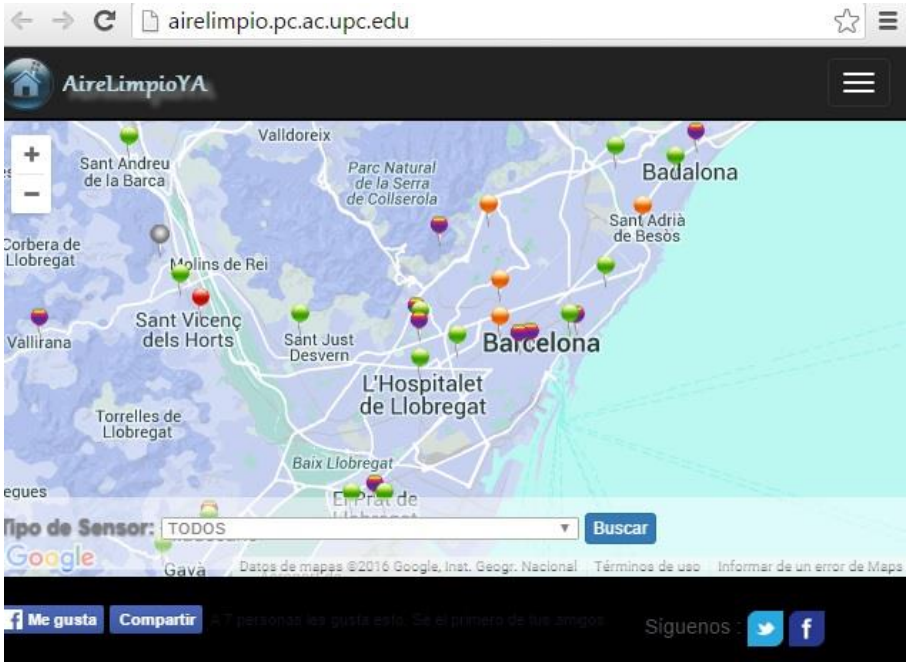
A continuació es detallen dos parts importants destacables dels projectes anteriors que són solucions inicials ja existents del problema:

- ***Commsensum***[1] és la plataforma que obté la informació dels sensors periòdicament, la guarda a la seva base de dades en format estàndard mysql, i la publica en diversos formats RDF seguint la ontologia definida. La part d'emmagatzematge i publicació d'informació és un mòdul separat que actua com a API que està desenvolupat amb Node.js, un llenguatge que permet rebre peticions de manera asíncrona, pel qual el converteix en el llenguatge perfecte, ja que permet rebre nombroses peticions tant per guardar informació com per obtenir-la. A continuació a l'imatge1 es pot veure de manera gràfica el seu funcionament en més detall. A l'esquerra està la capa de dades, on s'obté la informació de diferents fonts, tant de repositoris públics com de sensors. Després converteix la informació a un format adequat i el guarda a la plataforma. Finalment a la part inferior podem observar com l'aplicació final absorbeix la informació de la plataforma a través de diferents tipus de RDF.



Imatge 1: Arquitectura Commsensum

- ***AireLimpioYa***[3] és la aplicació que mostra les dades que obté de la plataforma Commsensum en un mapa de google map, mostrant els nivells de qualitat d'aire en cada sensor. A la imatge a continuació es pot observar la seva interfície, on els punts de colors són els sensors, i on cada color significa el grau de pol·lució.



Imatge 2: Interfície aplicació AireLimpioYA

Pel que fa plataformes que guarden la informació de diferents sensors i la mostren més o menys en temps real n'existeixen varies, però cap d'elles utilitza l'estàndard RDF, i tampoc te definit una ontologia. Aquestes plataformes el objectiu que tenen és generar diners i per això

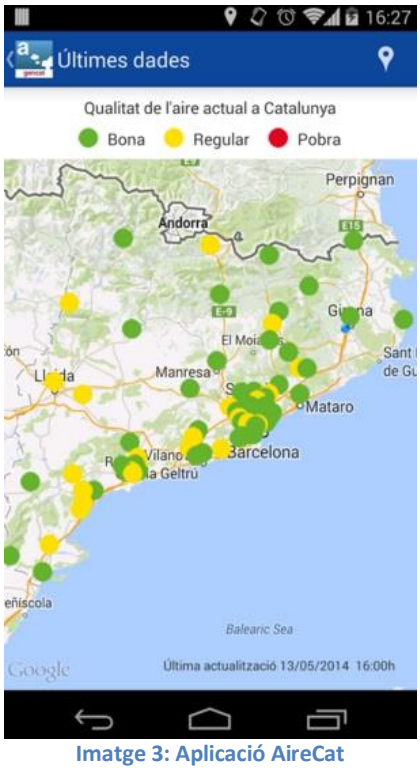
la majoria dels clients tenen la seva informació de manera privada. El que es pretén és tenir una plataforma de codi obert pública per tothom, que qualsevol persona pugui accedir al codi font i fer-ne bon ús.

Per exemple, una de les plataformes que existeix és ***thingspeak***[4] que et permet guardar fins a 8 valors d'un mateix sensor en text pla en un mateix canal de manera gratuïta i després mostra una gràfica. Un altre exemple és ***thethings.io***[5] que té unes funcionalitats semblants a la plataforma anteriorment comentada. Per tant, només hi ha una plataforma que s'adeqüi per complet als objectius, Commsensum.

A continuació, un altre dels objectius és crear un mòdul que extraurà la informació de certes fonts i les guardarà a la nostre plataforma. Aquest mòdul l'hem anomenat crawler per l'ús que tindrà. S'hauran de desenvolupar aquests dos mòduls, un per cada ciutat, des de zero.

En quant a aplicacions que mostrin els resultats de qualitats d'aire de la ciutat i de l'estat de les estacions de bicicletes n'hi ha varies, però cap d'elles mostra les dues informacions conjuntament. És a dir, només hi ha aplicacions o webs que mostren la informació per separat.

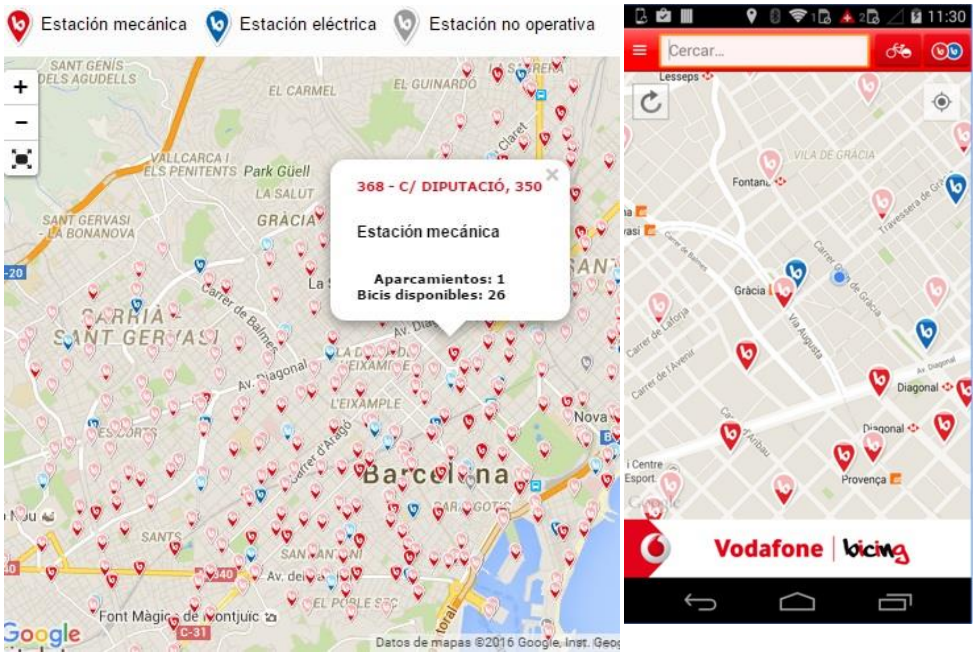
En el cas de la pol·lució per exemple la web AEMET[9] mostra la informació d'alguns sensors que analitzen la qualitat de l'aire. Un altre exemple és la aplicació ***AireCat*** que mostra la qualitat de l'aire de les estacions a nivell de Catalunya.



Imatge 3: Aplicació AireCat

[9] AEMET (Agencia Estatal de Meteorologia):

En el cas de l'estat de les bicicletes a Barcelona existeix la web oficial de ***Bicing*** la qual mostra un mapa amb l'estat de les estacions, inclús mostra les elèctriques. També existeix una aplicació Android oficial que mostra la mateixa informació.



Imatge 4: Web i aplicació oficial Bicing

Per tant, no hi ha cap aplicació ja existent que realitzi les funcionalitats definides com a objectius anteriorment. Així doncs, caldrà realitzar una aplicació web partint de l'aplicació desenvolupada millorada per un company que esta fent el projecte conjuntament. Aquesta aplicació mostrarà l'estat de la qualitat de l'aire de manera més clara en un mapa. El meu objectiu serà afegir la capa de l'estat de les estacions de bicicletes.

2.2. Abast

En un projecte gran amb moltes ambicions, si no es defineix un abast apropiat, pot resultar un fracàs. Delimitar fins a on arribarà el projecte serveix per tindre una visió més realista i a la vegada assegurar-te de la seva finalització. També és important analitzar i definir tant els possibles obstacles que poden sorgir, com les metodologies de treball que s'utilitzaran.

2.2.1. Abast del projecte

Podem identificar quatre punts clau que defineixen l'abast del projecte.

- Recol·lectar informació de forma periòdica del servei de bicicletes de les ciutats Barcelona i Saragossa, i guardar-la a la plataforma de forma ordenada. Hi haurà un

procés que farà de crawler, recol·lectant la informació especifica de les diferents fonts de dades obertes, parsejant el seu contingut, i guardant-la a la BBDD.

- Dissenyar o adaptar una ontologia ja creada per guardar la informació. Caldrà buscar si n'hi ha d'existents, i en cas afirmatiu, adaptar-la al projecte. En cas negatiu caldrà crear-la des de zero.

- Distribuir la informació obtinguda públicament amb format RDF/XML i RDF/JSON a través d'una API REST. Tota la informació que interessi recol·lectada a la BBDD caldrà distribuir-la a través de diferents URLs que es definiran a mida que avanci el projecte. Caldrà que aquesta informació estigui sempre actualitzada a la ultima versió disponible.

- Desenvolupar una web responsiva que mostri la informació tant de la qualitat de l'aire com de l'estat de les estacions de bicicletes de les ciutats de Barcelona i Saragossa. La aplicació web s'ha de mostrar correctament tant en dispositius mòbils com en ordinadors de sobretaula. Això s'aconseguirà utilitzant la llibreria Bootstrap i aplicant alguns estils.

2.2.2. Obstacles

Si els projectes anteriors no estiguessin suficientment clars i no es pogués aprofitar res de codi, llavors, s'haurà de dedicar molt més temps del previst al seu desenvolupament. També podria ser que l'anàlisi del codi fos contraproduent, perdent més temps analitzant-lo que el que es tardaria fent-lo des de zero, i això implicaria no tenir prou temps per realitzar tots els objectius del projecte. Aquests problemes podrien ser resolts treballant en equip i sent el més objectiu possible, analitzant en detall l'estat del codi i la seva possible reutilització.

El crawler actual està configurat perquè obtingui informació de dos ciutats i en format JSON. Quan es vulguin afegir més ciutats d'Espanya o d'arreu del món, s'haurà de tenir en compte que només funcionaria si utilitzen el format correcte pel que s'ha dissenyat inicialment. Aquest obstacle es pot solucionar desenvolupant un procés per format, i canviant la configuració perquè es pugui especificar el format de la font d'informació.

Un altre obstacle derivat del anterior cas, podria ser que hi haguessin dos fonts d'informació amb un mateix format, per aquestes tinguessin diferencies substancials pel que una d'elles es convertís en incompatible. Per exemple, si tinguéssim un fitxer JSON amb estructura de dades distribuïda d'un altre manera de la pre-definida, llavors s'hauria de desenvolupar un procés específic per aquest nou fitxer que difereix de la resta. A la llarga si no s'estandarditza la manera en la que es publiquen les dades, podria resultar en inviable desenvolupar una aplicació que agregues dades de múltiples fonts amb diferents formats.

Les dades de mobilitat s'actualitzen cada dos minuts, el que implica que el crawler ha de ser capaç de processar e inserir les dades a la plataforma en menys temps del que tarden a canviar les dades. De moment és possible fer-ho amb la informació de dues ciutats, ja que parlem de

2380 insercions, però quan es configurin més farà falta replantejar si es solucionaria amb més Hardware, o bé si faria falta redissenyar-lo, és a dir, per exemple distribuir la feina en paral·lel.

A part d'augmentar la carrega del processament de les dades, també creix el volum de dades que gestionarem. Al sistema actual fa servir una base de dades relacional que a mida que creixin les dades donarà pitjor rendiment, i no és permetrà escalar. Per solucionar-ho hauríem de pensar en utilitzar una base de dades noSQL, un tipus que està dissenyat per guardar grans volums de dades i la qual permet escalar horitzontalment sense problemes.

El fet de gestionar moltes dades que canvien constantment pot ser un problema a llarg termini quan la plataforma tingui que actualitzar milions de nodes cada dos minuts. La plataforma no estarà preparada per a processar tot aquest gran volum de dades suficientment ràpid. És crucial poder recopilar i publicar la informació a temps, sinó es perd tot el sentit que té la informació en temps real. Per una banda es podria renunciar a cert tipus de dades els quals no sigui tant important tenir les ultimes versions de les dades. Per altre banda, si es vol mantenir les dades en temps real s'hauria de redissenyar la plataforma, per començar utilitzar una base de dades noSQL.

2.2.3. Metodologia i rigor

En aquest apartat es descriuen les metodologies de treball, les eines de treball que s'utilitzaran, i quins mètodes de validació i seguiment es faran servir.

2.2.3.1. Metodologia de treball

Utilitzar una bona metodologia de treball és crucial perquè el projecte vagi avançant adequadament aprofitant al màxim el temps. És important fer ús de bones eines de treball que augmentin la productivitat i el seguiment, fent possible el desenvolupament per complet a temps treballant amb una planificació realista.

En aquesta metodologia es defineixen tres punts importants a tenir en compte:

Definir la feina, és a dir, detallar en blocs les diferents tasques que s'hauran de realitzar, i com més petites siguin les tasques millor. Implica consultar amb els directors del projecte perquè validin que aquestes són correctes i que no s'està planejant feina en va. Aquest punt és molt important per optimitzar al màxim el temps i ser productiu.

Crear un diagrama de les tasques especificant els deadlines. Aquest serà general, però, també es definiran les tasques que s'han de realitzar diàriament o almenys setmanalment, especificant que s'ha de fer cada dia.

Definir etapes les quals els directors del projecte validaran el progrés fet fins aquella tasca, evitant realitzar feina en va. No tot es pot planificar o predir amb certesa, és necessària la

presencia d'almenys una persona experta que pugui orientar si en una de les tasques sorgeix algun problema inesperat o si no és possible avançar.

2.2.3.2. Validació i seguiment

Cada setmana (esporàdicament cada dues setmanes) s'aniran fent reunions de seguiment amb els directors del projecte i amb els altres alumnes que fan el projecte relacionat. En aquestes reunions cada membre exposarà els seus avenços i els seus dubtes, i després es faran suggeriments per millorar-lo. Aquestes reunions seran molt importants, ja que aquestes ajudaran a validar l'estat del projecte en tot moment portant un seguiment adequat d'aquest. Així doncs cada membre també podrà veure en quin estat es troba el seu projecte i els dels demés, perquè cal recalcar que hi haurà parts comunes que seran desenvolupades pels diferents membres.

2.2.3.3. Eines de treball

La utilització d'unes bones eines de treballa facilitarà el correcte desenvolupament del projecte i per això s'ha decidit utilitzar les següents:

Asana és una eina que permet gestionar projectes a través d'una web. En cada projecte es poden assignar seccions i tasques establint prioritats i deadlines. També es pot assignar a diferents membres del grup a una certa tasca.

Google drive és una eina ja molt coneguda. Nosaltres la utilitzarem per compartir informació comuna creant una carpeta compartida. Aquesta eina també permet visualitzar la majoria de les dades sense necessitat de descarregar-les. Probablement serà utilitzada també per generar documentació i fer la presentació.

Bitbucket és un repositori de codi que utilitza github. Permet tenir un repositori com a privat de manera gratuïta. Aquesta eina la utilitzarem per el desenvolupament, tenint el codi a la xarxa en tot moment. Al finalitzar penjarem el codi de manera publica a github si escau.

Slack és una eina que permet utilitzar un xat de manera gratuïta entre els diferents membres dels projectes. Ens permetrà parlar en qualsevol moment entre nosaltres de manera rapida i àgil. Aquesta és la eina que utilitzarem més.

2.3. Planificació temporal

El projecte s'ha iniciat al 15 de Febrer i té una duració estimada de quatre mesos i mig, acabant a finals de Maig el desenvolupament de l'aplicació web. Seguidament es realitzarà l'ultima fase on es redactarà la memòria final, es farà la documentació i es prepararà la

presentació finalitzant a finals de Juny. Si alguna de les fases s'acaba abans de lo planejat, es podran realitzar millores detallades més endavant.

2.3.1. Definició tasques

A continuació es llisten les diferents tasques a realitzar amb el temps estimat:

Fase	Temps dedicat (h)
Fase 1: Gestió del projecte	**70**
Anàlisis del problema	20
Requisits i riscos	10
Planificació temporal	10
Planificació de costos	10
Sostenibilitat	10
Presentació	10
Fase 2: Desenvolupament del Crawler	**230**
Investigació de les tecnologies	35
Definició d'ontologia	15
Adaptació Commsensum	30
Implementació Crawler Barcelona	80
Implementació API REST	40
Implementació Crawler Saragossa	30
Fase 3: Aplicació web	**10**
Fase 4: Redacció de memòria, documentació i defensa	**90**
Total	**400**

Taula 1: Temps per tasca

Fase 1: Gestió inicial del projecte: 70h

Prioritat: **Alta**

La primera fase es farà un primer estudi del problema, analitzant els requisits necessaris i els possibles riscos. També s'elaborarà una planificació tant temporal com de costos. Addicionalment al final d'aquesta fase es farà un estudi de la sostenibilitat. Aquesta fase no te cap dependència.

Fase 2: Desenvolupament del Crawler: 230h

Prioritat: **Alta**

La següent fase és la més important i té una estimació de més de la meitat del temps total del projecte. En aquesta no només s'inclou el desenvolupament dels dos crawlers recol·lectors d'informació, sinó que també farà falta un nombre considerable d'hores per investigar les

tecnologies que s'utilitzaran, definir una bona ontologia i adaptar-ho tot a la aplicació ja existent Commsensum. També s'haurà de dissenyar el format més adequat a l'hora de compartir la informació de forma pública a la API REST. Aquesta fase té com a dependència la primera fase (fase 1).

Fase 3: Aplicació web: 10h

Prioritat: **Baixa**

L'últim objectiu és crear una aplicació web que mostri tant la informació de la qualitat d'aire com de l'estat de les estacions de bicicletes de la ciutat. S'espera partir d'una aplicació ja funcional realitzada per un altre alumne que està fent el projecte a la vegada en el mateix grup d'investigació. Per això queda justificat el nombre d'hores, que és poc comparat amb les altres fases. Aquesta fase té com a dependència les anteriors fases (fase 1 i 2), ja que sense el seu correcte funcionament no es pot realitzar la aplicació.

Fase 4: Redacció de memòria, documentació i defensa: 90h

Prioritat: **Mitjana-Alta**

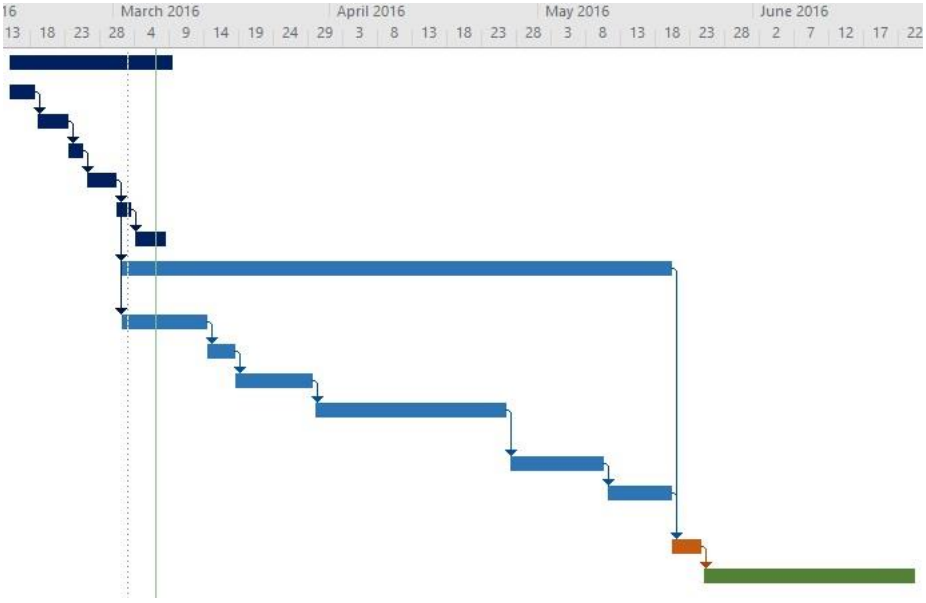
Aquesta tasca s'anirà realitzant una mica en cada fase, i la resta, la memòria final i la preparació de la defensa, al final del projecte. Per això no està detallat com una fase, perquè estarà integrada en cada una de les altres fases. No existeix cap dependència en aquesta tasca.

El temps total estimat de totes les fases és d'unes 400 hores, que si hi ha una dedicació diària d'unes 4 hores durant els dies hàbils, té més o menys una durada de 5 mesos.

A continuació és mostren la llista de tasques especificada i el diagrama de Gantt resultant. Tal com estava calculat, el projecte finalitzaria el 24 de Juny. A la taula de tasques (Imatge 5) les dues columnes de la dreta estan especificant quan iniciaria cada tasca i quan finalitzaria. Per deixar-ho més clar, al diagrama (Imatge 6), cada color és una fase diferent. Cal recalcar que la fase 2 s'inicia en paral·lel, conjuntament amb la tasca Sostenibilitat que la precedeix, després de la finalització de la planificació de costos.

	Task Name	Duration	Start	Finish
1	Fase 1: Gestió del projecte	70 hrs	Mon 15/2/16	Wed 9/3/16
2	Anàlisis del problema	20 hrs	Mon 15/2/16	Thu 18/2/16
3	Requisits I riscos	10 hrs	Fri 19/2/16	Tue 23/2/16
4	Planificació temporal	10 hrs	Tue 23/2/16	Thu 25/2/16
5	Planificació de costos	10 hrs	Fri 26/2/16	Tue 1/3/16
6	Sostenibilitat	10 hrs	Tue 1/3/16	Thu 3/3/16
7	Presentació	10 hrs	Fri 4/3/16	Tue 8/3/16
8	Fase 2: Desenvolupament del crawler	230 hrs	Wed 2/3/16	Fri 20/5/16
9	Investigació de les tecnologies	35 hrs	Wed 2/3/16	Mon 14/3/16
10	Definició d'ontologia	15 hrs	Mon 14/3/16	Fri 18/3/16
11	Adaptació Commsensum	30 hrs	Fri 18/3/16	Tue 29/3/16
12	Implementació Crawler Barcelona	80 hrs	Wed 30/3/16	Tue 26/4/16
13	Implementació API REST	40 hrs	Wed 27/4/16	Tue 10/5/16
14	Implementació Crawler Saragossa	30 hrs	Wed 11/5/16	Fri 20/5/16
15	Fase 3: Aplicació web	10 hrs	Fri 20/5/16	Tue 24/5/16
16	Fase 4: Redacció de memòria, documentació i defensa	90 hrs	Wed 25/5/16	Fri 24/6/16

Imatge 5: Taula de tasques de Gantt



Imatge 6: Diagrama de Gantt

2.3.2. Alternatives i pla d'acció

Si hi ha alguna desviació d'alguna de les tasques, provocaria que el temps planificat del projecte s'allargués aquest temps. Si això passa, que sol passar, s'espera una desviació no gaire elevada i la solució coherent per acabar-lo a temps serà dedicar més hores si la tasca o fase és prioritària, ja que s'ha planificat una dedicació de 4 hores diàries que podria augmentar en períodes puntuals. Si passa a la tercera fase, la del desenvolupament de la web, no es dedicarà més temps perquè no es una fase prioritària. En canvi si passa a la segona fase, caldrà dedicar més temps per tal d'acabar a temps.

De totes maneres hi haurà comunicació directe cada una o dues setmanes amb el director del projecte, una persona molt més experimentada, que ajudarà a prioritzar entre les diferents subtasques de cada fase. A la vegada, per les tasques conjuntes amb els altres dos membres que estan realitzant també el projecte en la mateixa àrea, és prendran decisions consensuades.

2.3.3. Recursos

En aquesta secció s'especifiquen els recursos necessaris per a la realització del projecte tant Hardware com Software.

2.3.3.1. Recursos Hardware

Els recursos Hardware que s'utilitzaran són les següents:

- Ordinador Dell + Pantalla 24'' per treballar des de l'oficina.
- Portàtil Asus A55V per treballar des de casa.
- Màquina virtual del departament per desenvolupar en producció.

2.3.3.2. Recursos Software

Les eines Software que s'utilitzaran són les següents:

- Microsoft Office 2016 + Project 2016
- Sublime text
- Github
- Python + Django Framework
- Adana
- Slack

- Bitbucket

2.3.3.3. Recursos Humans

Pel que fa els recursos humans, s'utilitzaran tres perfils clarament separats:

- Perfil de cap de projecte, l'encarregat de realitzar la primera fase i ajudar a les altres fases.
- Perfils d'analista i de programador que realitzaran les tasques de desenvolupament i documentació.

2.4. Gestió econòmica i sostenibilitat

En aquest apartat es farà una estimació econòmica, tenint en compte la planificació temporal especificada en el apartat anterior del total del projecte. S'especificarà el cost de recursos humans, de hardware, de software i altres detallats a continuació. També es calcularan les contingències.

2.4.1. Identificació dels costos

Primer de tot podem identificar el cost per personal que consta dels salaris que farien falta pels diferents rols. Cada rol té un preu per hora específic en el mercat laboral, i cada fase del projecte té associat una persona amb un rol específic. A continuació en les següents seccions s'especifiquen el cost dels recursos hardware de maquinaria necessari, de programari necessari i d'altres que engloba els costos del lloguer, de la llum, de l'aigua i de el internet.

2.4.2. Estimació dels costos

En el següent apartat es calcula en detall els diferents costos, i a l'últim apartat es calcula el cost final.

2.4.2.1. Recursos humans

A la següent taula es mostren els salaris associats amb els diferents rols. Els salaris estan basats en l'estudi aproximat de les remuneracions en l'àmbit de la tecnologia a Michael Page de l'any 2015[16], preus en brut.

Rol	Salari
Cap de projecte (C)	25 €/h
Analista (A)	19 €/h
Programador (P)	19 €/h

Taula 2: Salari de cada rol

A continuació a partir de les dades d'aquesta taula i de l'especificació del nombre d'hores a cada fase s'elabora el cost total en recursos humans.

Fase	Temps dedicat (h)	Rol	Cost
Fase 1: Gestió del projecte	**70**	**C**	**1.750 €**
Fase 2: Desenvolupament del Crawler	**230**	**A, P**	**4.370 €**
Fase 3: Aplicació web	**10**	**P**	**190 €**
Fase 4: Redacció de memòria, documentació i defensa	**90**	**A, P**	**1.710 €**
Total	**400**		**8.020 €**

Taula 3: Cost per fase

2.4.2.2. Recursos hardware

Els recursos hardware que s'utilitzaran estan detallats a la taula 5 a continuació. Es contempla un ordinador personal fixe per treballar a l'oficina, un portàtil per treballar des de casa i l'ús d'una màquina virtual senzilla gratuïta disponible pel DAC, i una màquina virtual més potent professional per el funcionament final del crawler. El preu de les possibles reparacions serà 0 € perquè tots els dispositius estan en garantia de 2 anys.

En la següent taula 4 es detallen les especificacions de la màquina virtual professional, servei que ofereix la companyia OVH. Com es pot observar és una màquina força potent tot i que no és necessària. Aquest servidor estarà virtualitzat i s'utilitzarà només una màquina virtual necessària.

Model	CPU	Core/Thread	Freq.	RAM	HD	Xarxa
HOST-32L	Intel Xeon D1520	4c/8t	2.2/2.6Ghz	32GB	4TB	1Gbps

Taula 4: Especificacions màquina virtual

El càlcul del cost de la màquina virtual[17] és el cost de la instal·lació més el cost per mes pel nombre de mesos, en aquest cas cinc. També es calcula l'amortització[18] de cada dispositiu. El total resultant sense IVA contant tot el hardware és el següent:

Producte	Preu	Vida útil	Cost	Amortització
Ordinador Dell	1.074,38 €	4 anys	1.074,38 €	107,44 €
Portàtil Asus A55V	495,87 €	4 anys	495,87 €	49,59 €
MV OVH	299,95 €	N/A	299,95 €	299,95 €
Total			**1870,2 €**	**456,98 €**

Taula 5: Cost recursos hardware

2.4.2.3. Recursos software

Els recursos software que s'utilitzaran incloent el càlcul de la amortització estan detallats a la taula següent. Es contempla una llicencia d'ús del software Microsoft Office 2016[19] de cinc mesos per la realització de tota la documentació incloent la memòria, presentació i documentació. Després també s'usarà el software Microsoft Project 2016 per realitzar la planificació i el seguiment del projecte. Finalment es detalla tot el software lliure gratuït que s'usarà per realitzar la resta de tasques. Cal recalcar que totes les actualitzacions de software són gratuïtes.

Producte	Preu	Vida útil	Cost	Amortització
Microsoft Office 2016	123,14 €	2 anys	123,14 €	24,63 €
Microsoft Project 2016	635,54 €	2 anys	635,54 €	127,1 €
Sublime Text, Github, Python, Django Framework, Adana, Slack, Bitbucket i Ubuntu	0 €	N/A	0 €	0 €
Total			**758,68 €**	**151,73 €**

Taula 6: Càlcul del preu dels productes de software

2.4.2.4. Altres

En aquest apartat es detallen els costos indirectes de la factura del consum dels dos ordinadors, del lloguer del local, de l'aigua i del internet.

El consum de l'ordinador fixe Dell i del portàtil, conjuntament amb el cost de la il·luminació del despatx on es treballa està detallat a continuació basat en la tarifa de la companyia Endesa[20].

Dispositiu	Consum	Preu per kWh	Consum total	Cost
Ordinador Dell	500W	0,1157 €	250 KwH	28,93 €
Portàtil Asus	300W	0,1157 €	165 KwH	19.09 €
Altres	200W	0,1157 €	100 KwH	11,57 €

Total	**515 KwH**	**59.59 €**

Taula 7: Càlcul del consum total

El cost del lloguer són 400 € mensuals d'una oficina de 25 metres quadrats que inclou tots els mobles necessaris (taules i cadires) i un lavabo. Així doncs el total del lloguer ascendeix a 2.000 €.

El consum en aigua és molt petit ja que només es consumeix a través del lavabo que te també una pica, amb un consum estimat de 12 litres diaris. El cost mensual són 12 € amb la companyia Aigües de Barcelona[21], i el total dels cinc mesos 60 €.

Addicionalment també s'ha contractat una línia d'internet de fibra òptica de Telefònica-Movistar[22] amb un cost total de 49.59 € cada mes, el que fa un total de 247,93 €.

Finalment, el total ascendeix a 2.367,52 € incloent els quatre costos detallats anteriorment.

2.4.2.5. Contingències i imprevistos

Un dels problemes que pot aparèixer és que s'hagi de dedicar més hores de les planificades, tal com està detallat a l'apartat desviacions. Això pot provocar un augment del cost total pressupostat degut a les hores extres de dedicació. S'ha establert una possible desviació de 60 hores, que suposa un 15% del temps total, amb un cost en el pitjor dels casos del rol de Cap de projecte amb salari de 25 €/h. Això dona un total de 1.500 € addicionals. El projecte acabarà a temps, tal com està planificat.

2.4.2.6. Cost total

A la següent taula està detallat el cost total de tots els recursos.

Detall	Cost sense IVA	Cost amb IVA
Recursos humans	NA	**8.020 €**
Recursos hardware	456,98 €	**552.95 €**
Recursos software	151,73 €	**183,59 €**
Imprevistos	NA	**1.500 €**
Electricitat	59.59 €	**72,1 €**
Lloguer	2.000 €	**2.420 €**
Aigua	60 €	**72.6 €**
Internet	247,93 €	**300 €**
Total	10844,5 €	**13.121,24 €**

Taula 8: Cost total

2.4.3. Control de gestió

Com ja s'ha comentat anteriorment, es pot donar el cas de que la planificació no es pugui dur a terme al peu de la lletra, com passa a la majoria de projectes, ja que apareixen imprevistos.

En cap cas faran falta més recursos ni de hardware ni de software. En el primer cas hipotètic podria ser que haguéssim d'augmentar el nombre d'hores a dedicar per a la fase de desenvolupament, fase complicada i el nucli del projecte, és a dir, hi hauria més costos en recursos humans. El segon cas podria ser que necessitéssim més temps i per tant, haguéssim de pagar el lloguer, electricitat, aigua i internet un mes més.

Per assegurar la finalització del projecte solucionant els dos casos detallats anteriorment, s'ha definit al pressupost un augment de 1.500 € en cas d'imprevist.

3. Sostenibilitat

En aquest apartat es plantejarà la sostenibilitat des de tres punts de vista: ambiental, econòmic i social. A l'últim apartat es detalla una taula amb la puntuació final del projecte que avalua amb una matriu la sostenibilitat.

3.1. Ambiental

En principi, en la fase de desenvolupament les úniques despeses que poden afectar són els costos indirectes de la corrent que es consumeix, ja que aquesta és creada amb uns processos de fabricació que normalment si deixen petjada al medi ambient.

El crawler ja desenvolupat, el qual obté informació sobre les estacions que prenen la mida de la qualitat de l'aire a Catalunya i a Madrid, serà un punt de partida a l'hora de desenvolupar-ne el nou, almenys per tenir idees. Ho podem entendre com una reutilització del crawler anterior. També es reutilitzarà la aplicació web que mostra les dades de les estacions que prenen mesures de la qualitat de l'aire.

Tipus	Consum (5 mesos)
AC/Calefacció centralitzada	**1000 kWh**
2 MV	**216.66 kWh**
PC	**108.33 kWh**
Il·luminació	**112 kWh**

Taula 9: Consums per elements

S'ha estimat un consum de 1433 kWh [23] en el desenvolupament, que inclou el consum d'un ordinador de sobretaula, dos màquines virtuals, la il·luminació i l'aire condicionat/calefacció centralitzada tal com es detalla a la taula superior. El que consumeix més és l'aire condicionat/calefacció centralitzada i no hi ha manera d'apagar-lo. Per reduir-ho es proposa treballar en un altre local, on no sigui obligatori l'ús d'aquest. Una altre mesura que podríem prendre es apagar totes les màquines virtuals que no s'utilitzen cada cop. També podríem utilitzar ordinadors de baix consum, com per exemple un Raspberry PI 3 que consumeix uns 10W. Sense reduir el consum i segons el factor de conversió de que 0.385Kg de CO_2 equivalen a un Kwh [24], obtenim un total de 551.705 KG de CO_2.

Si volgués realitzar el projecte de nou replantejaria quins recursos són totalment necessaris i prescindiria dels innecessaris. Probablement només faci falta un ordinador que pot ser de baix consum, una màquina virtual amb el hardware mínim i il·luminació suficient per treballar. Tota la resta podria ser prescindible.

El projecte si tot va bé tindrà una vida útil d'almenys un any, i el que fa falta és tant sols una màquina virtual en funcionament les 24h. Si el consum que té és de 200 KwH, generaria uns 77 KG de CO2, molt menor que el generat durant el desenvolupament.

Durant tota la vida útil del projecte es disminuirà la petjada ecològica, ja que amb l'aplicació web es pretén conscienciar els ciutadans perquè utilitzin menys el transport privat i més el transports públic sostenible com el Bicing a Barcelona. D'aquesta manera, si altres ciutats prenen exemple, es disminuiria encara més a nivell mundial. Així doncs, per cada 77 KG de CO2 que es generarien per cada servidor que utilitzi el crawler i mostri la web anualment, evitaríem que un cert nombre de persones utilitzin el transport privat, conscienciant-los amb les dades de l'aplicació.

És possible que si el projecte desenvolupat s'utilitza en un hardware sobredimensionat, aquest generi més CO2 del que hauria. No es pot controlar a on funcionarà el software, tant sols es pot informar als usuaris de amb quin hardware mínim hauria de funcionar correctament.

Per reduir l'impacte, la millor opció seria no utilitzar AC/Calefacció centralitzat. O utilitzar dispositius de baix consum o utilitzar el mínim hardware possible. En el nostre cas tenim un servidor virtualitzat, per tant, podem adequar el hardware de la màquina a les necessitats reals, podent baixar o pujar la potencia d'aquesta.

3.2. Econòmica

El temps dedicat a cada tasca és proporcional a la seva importància, perquè per exemple en la fase de desenvolupament s'ha planificat més de la meitat del temps total, perquè és molt important crear un molt bon crawler que llegeixi correctament la informació i estigui preparat per llegir-la en altres formats. És el nucli del projecte. També es dedica molt temps en la part final on s'ha de documentar el projecte perquè qualsevol persona pugui seguir el seu procés, col·laborar, etc.

En paral·lel s'estan realitzant quatre projectes que utilitzen la mateixa plataforma Commsensum. Tots els membres estem a la mateixa aula i en algunes parts es treballarà conjuntament. El primer projecte va sobre la realització d'una aplicació mòbil millorada de la actual airelimpioYA, on es mostrarà més informació més clara i serà responsiva. El segon projecte va sobre la recol·lecta d'informació sobre tots els sensors que monitoritzen la qualitat de l'aire d'Espanya, i alguns d'Itàlia i Àustria. El tercer va sobre la implantació de sensors de baix cost, analitzant la seva viabilitat, iniciat en algunes regions de Catalunya. A la vegada també s'està col·laborant amb una associació anomenada ***Ecologistas en acción*** a nivell nacional, i amb altres universitats d'Itàlia i Àustria en el segon projecte.

Pel que fa a l'estimació del cost total de la realització del projecte, no ha variat gaire respecte la planificació inicial. L'únic element que ha canviat ha sigut que enlloc d'utilitzar un servidor virtualitzat de gama baixa que costava uns 20€/mes, hem utilitzat un professional que costa

uns 60€/mes. El actual servei contractat és de més qualitat a part d'oferir molta més potencia. S'ha canviat a un servei millor pensant en un futur, per poder implementar noves millores i per no tenir problemes de rendiment almenys en el termini d'un any. Actualment aquest servei no s'utilitza en tota la seva totalitat, tant sols s'utilitzen les maquines virtuals imprescindibles. Així doncs, aquest canvi ha provocat un augment de 40€ mensuals per els 5 mesos, sumant un total de 200€ addicionals.

Tal com s'ha comentat a l'apartat anterior, una reducció en costos del desenvolupament es podria plantejar si utilitzéssim maquines de baix consum, per exemple Raspberry PIs 3, i si treballéssim en un local on l'aire condicionat i la resta d'elements generessin menys despeses. Un altre opció també seria utilitzar aquests dispositius o similars de baix consum en producció configurats com a clúster, ja que aquests són menys fiables i menys potents que un ordinador normal. El consum d'un ordinador genèric son 22kWh mensuals, on 5kWh són del monitor i 17kWh de la torre. Una torre de baix consum consumeix menys de la meitat, uns 7.2 kWh (suposant un consum de 10Wh de la Raspberry PI), aconseguint un 44% d'estalvi d'energia. Si aquests tipus de dispositius s'implantessin tant a la fase del desenvolupament com en els servidors de producció tindríem un estalvi considerable en la llum, i a la contractació del servidor virtualitzat. Caldria comprovar quines opcions ofereix el mercat i si són suficientment potents per les necessitats existents.

En l'apartat anterior es proposa l'ús de maquines de baix consum, com la Raspberry PI en configuració de clúster, per utilitzar-les en producció. Aplicant aquesta estratègia, si un servidor en producció virtualitzat consumeix uns 150Wh, si utilitzem quatre Raspberry Pi amb un consum total de 40Wh, ja tindríem un estalvi del 73%, tenint un hardware bastant semblant. S'ha de tenir en compte que aquestes maquines estaran enceses 24h, per tant és important utilitzar maquines que consumeixin el mínim possible. Però, el cost inicial seria d'uns 40€ per unitat, sumant un total de 160€. Per tant, com que el cost del servidor virtualitzat és de 60€ al mes, l'estalvi seria a partir del 3r mes.

El cost total de la vida útil es podria veure afectat si tenim en compte les actualitzacions i els possibles problemes que puguin sorgir, i que faci falta arreglar el software. Durant el projecte, l'actualització automàtica d'un paquet va provocar la fallada d'un dels mòduls, per tant, hi ha probabilitat de que torni a passar alguna cosa semblant. També s'ha de tenir en compte que és important tenir el software actualitzat si és possible, almenys amb els paquets que solucionen errors de seguretat o bugs. Per tant farà falta invertir més hores a una persona que haurà de portar el manteniment del servidor i del software associat, dedicant 8h per setmana, sumaria un total de 416 hores anuals. Si amb un perfil d'analista programador ja és suficient faria falta invertir 7.904€ més. Una altre opció seria que un membre del grup d'investigació s'encarregués també d'aquest manteniment, estalviant-nos contractar una persona addicional.

En el optimista cas de que l'aplicació tingués molt d'èxit i es comencés a utilitzar molt per molta gent, caldria replantejar aspectes arquitectònics de l'aplicació. Implicaria dedicar més personal dedicat a la millora del software, i contractar un servidor virtualitzat millor o comprar més maquines. Podria ser que la contractació de tots aquets recursos fos inviable. També cal tenir en compte que sense aquesta dedicació de recursos extra, el projecte no compliria

l'expectativa de tots els nous usuaris i deixaria d'utilitzar-se, convertint el projecte en un fracàs. Per altre banda, també podria passar el contrari, que alguns usuaris o institucions interessats pel software volguessin col·laborar, i entre tots poguéssim avançar més ràpid cap a un software millor.

3.3. Social

El projecte es troba dins del món de les Smartcities i del IoT, àrees cada cop més importants. El que es pretén és impulsar la ciutat de Barcelona perquè cada cop sigui més smart i més mòbil, és a dir, que cada cop es generi més informació open data rellevant i útil de la ciutat, i que aquesta sigui aprofitada per aplicacions de tercers que millorin la qualitat de vida de tots els ciutadans. Per marcar un abast realista es va decidir centrar-se en la mobilitat, més en concret al servei d'estacions de bicicleta de les ciutats. A la vegada també es pretén impulsar l'ús de tecnologies estàndard de representació de dades com és el RDF.

En un primer anàlisis inicial previ a la finalització del projecte, podem dir que si que hi ha una necessitat real d'aquest producte, tot i que el projecte només serà un petit pas més, amb el qual s'aconseguirà obtenir una capa d'informació de mobilitat de la ciutat. Aquesta serà guardada de manera entenedora a la plataforma ja existent i permetrà crear aplicacions que depenguin d'aquesta informació, que funcionaran sense pràcticament cap canvi en diferents ciutats que tinguin el mateix tipus d'informació, i que utilitzin el mateix estàndard. Quan estigui finalitzat el projecte, serà possible utilitzar l'aplicació creada en diferents ciutats d'Espanya, només modificant el crawler que llegeix la informació, la transforma i la guarda a la plataforma.

Un cop ja finalitzat el projecte, l'impacte que ha tingut la realització del projecte sobre mi ha estat més de la que hagués pogut imaginar. Per començar desconeixia iniciatives que treballaven en l'àmbit del "internet de les coses" i de la pol·lució aquí a la universitat. El fet de saber que hi havia un grup darrera amb objectius de millorar el planeta em va donar més ganes de treballar amb ells. La oportunitat de col·laborar amb el meu projecte, en un projecte comú, m'ha confirmat que vaig prendre una bona decisió en triar-lo. Per altre banda, analitzar la gran quantitat de CO2 que es genera només en la producció de l'electricitat necessària per realitzar el projecte, em va fer veure que les petites accions són importants, i que com a futur informàtic hauré de prendre decisions que sense dubte involucraran en més o menys mesura la sostenibilitat.

Les persones de peu que es preocupen per el medi ambient seran els més beneficiats d'aquest projecte. L'aplicació final és el primer pas cap el desenvolupament d'una aplicació totalment funcional i útil per veure quines zones són les més contaminades, i per tant quina ruta hauria de seguir, tenint en compte l'estat de les estacions de bicicletes. També els professionals que vegin com a una bona alternativa aquesta iniciativa i vulguin implementar la seva pròpia versió a la seva ciutat o país. Si hi ha més ús de bicicletes probablement l'empresa que ofereix el servei també es vegi beneficiada. Pel grup d'investigació aquest projecte podria demostrar les possibilitats que existeixen i obrir portes a més camins, per tant, tant si es continua o no amb aquest s'estaria beneficiant també. En general, totes les persones que utilitzin aquesta

aplicació i usin les bicicletes com a transport, estaran més sanes i això farà reduir el nombre de malalts.

El resultat final ha sigut més del planejat inicialment. Per exemple s'han desenvolupat millores a la API REST que també beneficiaran a altres projectes. També s'ha analitzat la possible utilització d'una base de dades no relacional enlloc de la relacional, quan no hi havia necessitat de fer-ho. Els resultats han sigut més que acceptables, i es veu que realment que el que he estat fent m'agradava i hi posava il·lusió. Considero que és molt important posar passió o il·lusió, et permet treballar més a gust i obtenir millors resultats finals.

Després d'analitzar-ho durant dies no s'ha trobat cap segment de la població que pogués veure's afectat pel projecte. Però, podria haver dependència d'ús per les persones que només confiessin amb aquesta aplicació, i si pel que fos deixes de funcionar en algun moment donat, important per ells, deixarien de confiar en aquesta aplicació. Aquest cas es pot donar, però no amb molta freqüència, i existeixen altres fonts d'informació que podrien consultar si aquesta situació passés.

3.4. Puntuació informe de sostenibilitat

En aquest apartat es detalla la matriu de sostenibilitat de la part corresponent, és a dir, la primera columna corresponent a PPP[10], que delimita la finta inicial, és a dir, inclou la planificació, el desenvolupament i la implantació del projecte. Les altres dues columnes engloben la totalitat del projecte incloent la vida útil i els riscos inherents. En els apartats anteriors s'ha justificat la nota en cada una de les nou caselles, obtenint una puntuació final de 67 sobre 90. Podem afirmar doncs que el projecte és prou sostenible, i que s'ha tingut en compte els tres àmbits, tant ambiental, com econòmic i social.

	PPP	Vida útil	Riscos
Ambiental	Consum del disseny	Petjada ecològica	Riscs Ambientals
	7	16	-1
Econòmic	Factura	Pla de viabilitat	Riscos econòmics
	8	16	-2
Social	Impacte personal	Impacte social	Riscos socials
	8	16	-1
Rang sostenibilitat	23	48	-4
	67		

Taula 10: Matriu sostenibilitat

[10] PPP: Projecte posat en producció.

4. Tecnologies

Cada cop hi ha més tecnologies per triar per realitzar el mateix i és més difícil prendre decisions clares, és a dir, que hi ha múltiples tecnologies diferents que serveixen igual per un mateix objectiu. Per seguir el mateix format el qual han sigut desenvolupades les anteriors versions s'ha decidit triar les mateixes tecnologies que segueixen sent competents i viables.

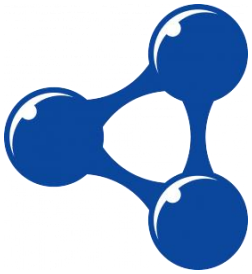
La plataforma i la web van ser desenvolupades amb el framework de ***Python Django***, i Node.js per la ***API REST***, el mòdul de consulta i emmagatzematge d'informació. S'ha estudiat i per el crawler s'ha decidit utilitzar Python. La plataforma utilitzarà ***Mysql*** com a base de dades, tindrà definida una ontologia en llenguatge ***RDF Schema***. La informació serà distribuïda amb la ***API REST*** ja desenvolupada, en format ***RDF/XML*** i ***RDF/JSON***. Pel que fa la aplicació, es farà una web responsiva utilitzant la combinació del framework ***Django*** en ***Python*** i de la llibreria de ***Bootstrap***, que mostrarà la informació en un mapa utilitzant la ***API de google maps***. A continuació es descriuen les tecnologies en més detall.

4.1. Web semàntica

Aquest concepte[6] engloba un conjunt de activitats que es realitzen al W3C[11] amb l'objectiu de crear tecnologies estàndard per la publicació de dades llegibles per aplicacions informàtiques. La manera en que ho realitza és inserint metadades semàntiques i ontologies a la web. L'objectiu d'aquesta informació extra és descriure el contingut, el significat i/o la relació entre les dades de manera que una màquina pugui entendre millor de que es tracta. Així doncs l'objectiu és millorar l'actual web ampliant la interoperabilitat entre sistemes informàtics, els quals actuaran independentment sense la interacció d'una persona. Això s'aconsegueix seguint les directives que es defineixen a Linked Data. Aquest consorci defineix els estàndards de OWL[12] i RDF[13] que s'expliquen a continuació.

[11] World Wide Web Consortium: Es dedica a treballar per desenvolupar i promocionar estàndards per la WWW.
[12] Web Ontology Language: Dissenyay per ser utilitzat per aplicacions que necessiten processar el contingut de la informació en lloc de presentar-la al éssers humans.
[13] Resource Description Framework. És un model de dades per la web semàntica.

4.2. Linked data



El Web esta ple de dades, en formats com Pdf, CSV, Excel i molts més. Aquestes dades estan enllaçades entre elles a diferents documents HTML o similars. Així doncs, el model actual ja permet enllaçar dades, però aquest té una limitació molt important, tota aquesta informació està dissenyada perquè ho consumeixi una persona. Normalment , una ordinador si vol llegir aquesta informació necessita eines especials, i quasi sempre la interacció d'una persona. És molt complexa o inviable automatitzar l'accés, recerca i reutilització d'aquest tipus de dades.

El que proposa aquest nou model de compartir les dades és que es segueixin els següents consells:

- Les dades són accessibles al web (són dades obertes)
- Les dades són accessibles en formats estructurats de dades llegibles per maquines (per exemple Excel)
- Les dades són accessibles en un format no propietari (per exemple CSV)
- Les dades són publicades seguint algun dels estàndards de dades obertes del consorci W3C (per exemple RDF)
- Les dades a part, estan enllaçades amb dades d'altres persones.

Quan aquestes normatives es compleixin per totes les dades del web, estarem en una nova web en la qual qualsevol ordinador podrà buscar informació automàticament sense interacció humana, entenent el significat de les dades i la relació entre elles.

4.3. OWL

OWL[7] és un llenguatge que pertany a la web semàntica que serveix per definir ontologies o complexes per descriure qualsevol cosa, ja sigui un sol o grup de conceptes. Permet definir la relació entre els elements que la formen. L'objectiu d'aquest llenguatge és permetre publicar i compartir dades usant ontologies al web. Actualment es troba a la versió 2. Conjuntament amb el model de dades RDF, permet definir vocabularis a través dels elements entre d'altres.

La plataforma web Commsensum té definida una ontologia pròpia per gestionar totes les dades dels sensors de qualitat d'aire. Algunes de les dades que es publiquen segueixen aquesta estructura. Pel projecte no ha fet falta modificar-la i s'ha reutilitzat ja que el concepte era molt semblant.

4.4. RDF

RDF[8] és un model de dades estàndard que permet seguir els consells de Linked Data, i que serveix per compartir informació a la web de manera semàntica. Cada element d'informació està representat amb una tripleta, és a dir, tres parts. El subjecte és on va la URI de la dada, el Predicat és on va l'enllaç al tipus de dada en si o a un altre, i l'objecte que és el valor d'aquesta dada. A continuació es mostra un exemple:

<http://example.com/my_temperature_data> rdfs:label "Temperature observations";

El primer element és el subjecte que representaria la URI probablement amb les mesures de temperatura. El segon element és el predicat que representa quin tipus de dada és. El tercer element és l'objecte, és a dir, el valor de la dada. Així doncs aquest arxiu de mostra representaria la etiqueta d'una URI que sembla que enllaça amb les dades de temperatura.

Aquest format facilita la fusió de dades encara que les esquemes de dades siguin diferents, i suporta la evolució de les esquemes de dades en el temps sense requerir que els consumidors de les dades canviïn. La utilització d'aquest model permet barrejar informació estructurada i semiestructurada, i posteriorment compartir a través de diverses aplicacions. Tota aquesta estructura de dades forma un graf on es representarien la relació entre els diferents recursos representats en nodes.

També existeix una extensió, el RDFS[14][9], que és un llenguatge de propòsit general que serveix per representar vocabularis RDF a la web. Existeixen altres llenguatges que serveixen per representar també les dades de manera semàntica, però aquest és el que més s'utilitza.

Una gran part de les dades de la plataforma de Commsensum es publiquen amb aquest model, amb la combinació de XML, ja que aquest model de dades pot anar amb altre llenguatges com el JSON.

4.5. JSON

JSON[15] és un format lleuger d'intercanvi de dades, fàcil de llegir i escriure. També és fàcil de generar i parcejar per maquines[10]. És un format de dades completament independent del llenguatge que l'utilitzi, podent ser utilitzat per nombrosos llenguatges com C, C++, Java, Javascript, Python, etc. Aquest format a demés permet definir col·leccions de parelles de nom i valor, i llistes de valors, representades com un array, vector o llista.

[14] RDF Schema: Permet representar vocabularis usant l'estàndard RDF.
[15] Javascript Object Notation: És un format d'intercanvi de dades.

El crawler desenvolupat utilitzar aquest format per l'arxiu de configuració, i per empaquetar les dades que s'envien a la API REST. La aplicació web també l'utilitza per la gestió de les etiquetes del mapa de google maps.

4.6. HTTP

HTTP[16] és un protocol estàndard, definit per l'IETF[17] i el W3C, de la capa d'aplicació que serveix per l'intercanvi de documents d'hipertext i multimèdia al web[11]. La versió bàsica no està xifrada però existeix la versió segura millorada que utilitza SSL anomenada HTTPS.

Aquest protocol funciona vinculant les peticions amb les respostes, utilitzant un model client-servidor. L'estàndard disposa de les peticions de GET per obtenció d'un recurs en concret, de HEAD que funciona igual que el GET però retorna el contingut sense el body, i altres com el POST, DELETE, OPTIONS i CONNECT. Tots els navegadors utilitzen aquest protocol per llegir i mostrar les pagines web. També s'utilitza a les implementacions d'API REST, on les comunicacions es realitzen amb aquest protocol de manera personalitzada.

La API REST de la plataforma utilitza aquest protocol per publicar informació, i per rebre peticions d'insercions de dades. La aplicació web està en un servidor web, que rep peticions HTTP.

4.7. HTML

HTML[18] és un llenguatge basat en etiquetes utilitzat en les pàgines web[12]. Conjuntament amb Javascript i CSS[19] permet crear interfícies d'usuari en format pàgina web o per dispositius mòbils. Els navegadors web interpreten aquest llenguatge i el renderitzen mostrant la pàgina web. La ultima versió és la 5, en la que s'han canviat i afegit noves etiquetes, atributs, normatives , funcionalitats que ajuden a afegir més semàntica a les pàgines web. Les plantilles de la pàgina web estan fetes amb aquest llenguatge com a base.

[16] HyperText Transfer Protocol: És un protocol de comunicació usat en el WWW.
[17] Internet Engineering Task Force: Organització internacional creada a EE.UU. oberta a la normalització de diverses àrees d'internet, com la seguretat, l'encaminament i el transport.
[18] Hypertext Markup Language: Llenguatge d'etiquetes orientat a la creació de pàgines web.
[19] Cascading Style Sheets: És un llenguatge orientat a definir i crear estils a documents HTML o XML.

4.8. Bootstrap

Aquest open source framework[13] permet crear l'estructura del front-end de pagines web de manera ràpida i fàcil. És senzill d'utilitzar i compatible per qualsevol dispositiu de qualsevol mida, permetent crear webs responsives que s'adapten a qualsevol resolució de pantalla. Conté arxius d'estil CSS, arxius Javascript i algunes fonts. S'utilitza afegint classes a les etiquetes HTML, i la llibreria aplica una sèrie d'operacions sobre aquestes. Permet donar estil a qualsevol etiqueta, ja sigui un botó, un formulari o una llista. La pàgina web utilitza aquesta llibreria per perquè actuï de manera responsiva.

4.9. Django

Django[14] és un framework web de Python open source i gratuït que facilita el ràpid i net desenvolupament d'una aplicació web. Dissenyat per desenvolupadors experimentats, evita problemes típics del desenvolupament web, i permet a un focalitzar-se en escriure el codi de la aplicació. Les tres principals característiques en les que destaca són la seva rapidesa, la seva seguretat i la seva escalabilitat, les quals el converteixen en una bona alternativa. La seva ultima versió és la 1.9 tot i que en el desenvolupament s'utilitzarà una versió més antiga. La plataforma web Commsensum està feta amb aquest framework, igual que la aplicació web que s'ha desenvolupat.

4.10. Node.js

Node.js[15] és un ràpid motor d'execució de Javascript utilitzat en el navegador google Chrome anomenat V8. Aquest utilitza un model basat en events i no bloquejant d'entrada/sortida, és a dir, asíncron, que el converteix en molt lleuger i eficient. Utilitza el ecosistema de gestió de paquets NPM, el open source més utilitzat al món. El que el diferencia de Javascript és que aquest s'executa en el servidor i no pas en el client com passava antigament. Una altre característica important és que va ser dissenyat per ser un llenguatge altament escalable, el que el converteix en un llenguatge perfecte per la API REST.

Express és un framework de Node.js que facilita la seva utilització com a servidor web, és a dir, permet utilitzar el protocol HTTP per defecte. És força senzill d'utilitzar i està molt ben incorporat.

Un terme molt de moda, el MEAN stack, és refereix al domini de la utilització de quatre tecnologies orientades a Javascript. M de MongoDB, E d'Express framework, A d'AngluarJS i N

de Node.js. Estan agrupats perquè totes ells estan relacionades i funcionen molt bé en equip. Per exemple MongoDB emmagatzema les dades en JSON (Javascript Object Notation), Express i Node.js fan de servidor web utilitzant el llenguatge Javascript, i intercanvien fitxers JSONs amb el front-end que usa AngularJS.

4.11. MongoDB

MongoDB és un tipus de base de dades noSQL, en la que no existeix la definició de taules amb limitacions i relacions entre atributs. Si que incorpora mecanismes similars per poder representar-ho, com per exemple anomena Collection a una taula. Aquesta base de dades, com sol passar amb les d'aquest tipus, ha sigut dissenyada per emmagatzemar grans volums de dades sense perdre rendiment a l'hora de consultar-les perquè és escalable horitzontalment. La informació és guarda en format document, és a dir, en JSON o BSON, lo que la converteix en una alternativa perfecte si pretens moure dades d'aquest tipus.

4.12. MySQL

MySQL és un gestor de bases de dades relacional, multiusuari, que usa el llenguatge SQL, molt popular gràcies a la seva velocitat d'execució de les consultes. Va ser creat el 1995 i actualment és el més utilitzat. Està desenvolupat en C i C++, és multiplataforma, pertany a Oracle i és compatible amb molts llenguatges com JAVA, Python, PGHP, etc. . A demes aquest software es pot utilitzar de forma lliure i gratuïta sota les condicions de la llicencia GPL. Així doncs és la base de dades que s'utilitza a la plataforma Commsensum i a la aplicació web, combinat amb el framework Django de Python.

4.13. API REST

API REST fa referencia a la combinació de ***Application Programming Interface***, que podem definir com a conjunt de subrutines, funcions i procediments que ofereix una certa biblioteca per ser utilitzat per algun altre software com una capa d'abstracció, i ***Representational State Transfer***, que és una arquitectura de programari pensada per sistemes distribuïts basats en hipermèdia, que normalment utilitza el protocol HTTP. Aquesta arquitectura té unes característiques:

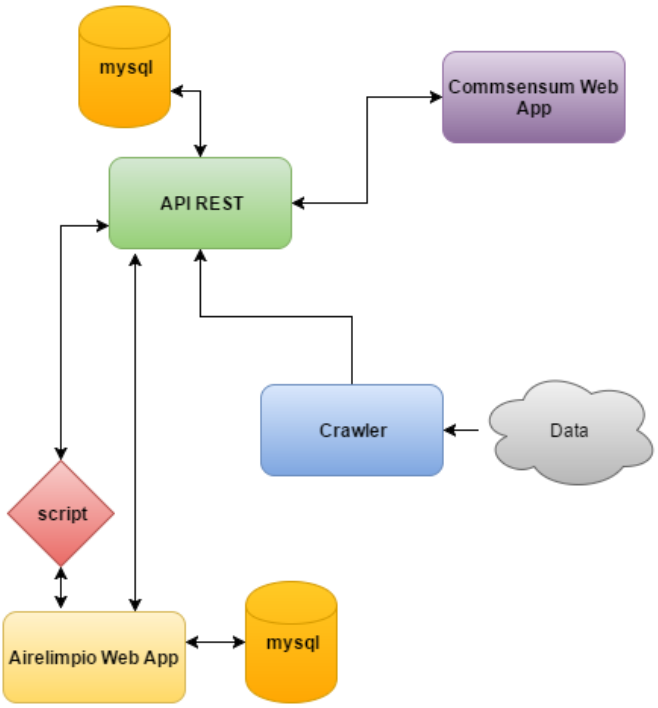
- Protocol client/servidor sense estat.
- Conte operacions ben definides com POST, GET, PUT i DELETE.

- Utilitza una sintaxi universal a través d'URIs
- Utilitza hipermèdia com el HTML o XML.

Últimament s'està posant de moda la utilització d'una API REST com a back-end en les pàgines web. Així s'aconsegueix una abstracció total de les dades a part de poder oferir serveis a altres aplicacions. Es podria canviar la base de dades que s'utilitza al back-end sense tindre que canviar les funcionalitats amb les que es comunicarien el front-end.

5. Desenvolupament

En el següent apartat es detallen els tres elements que s'han desenvolupat amb les seves respectives noves funcionalitats o millores. El primer element és el nucli del projecte, el crawler realitzat amb Python. El segon són les millores aplicades sobre la ja existent API feta en Node.js. L'últim element és l'aplicació web millorada també realitzada amb Python amb el framework Django. A continuació és pot veure l'arquitectura general, la qual s'explica en detall a cada element.

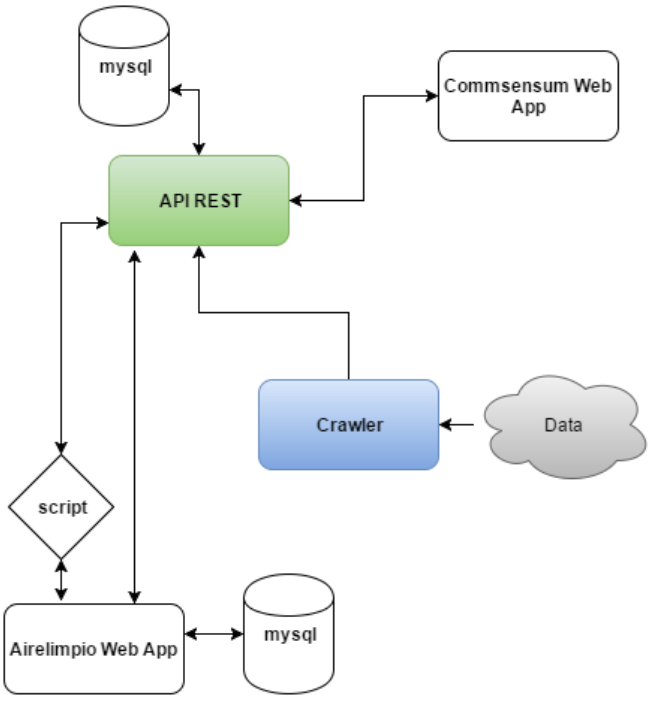


Imatge 7: Arquitectura general

5.1. Crawler

D'entre molts possibles llenguatges de programació aptes per la tasca que anava a realitzar el crawler s'ha triat el llenguatge Python, per la seva versatilitat, execució ràpida, de fàcil aprenentatge i amb mòduls suficients. Un altre factor també ha sigut que aquest s'utilitza força en el món IoT, en dispositius com per exemple una Raspberry PI, àrea d'interès personal.

A continuació es mostra un gràfic, on és pot observar l'arquitectura, on el crawler recopila dades d'internet, les processa i les inserta a la API REST mitjançant peticions HTTP.



Imatge 8: Crawler

Per tal de fer modular el codi, finalment ha resultat en tres fitxers:

- ***urbanCrawler.py***: Aquest és l'arxiu principal que utilitza els dos altres arxius per a realitzar les diferents execucions. Per defecte utilitza la inserció en bloc, la millora també implementada a la API. Permet dos modes d'execució, el normal especificant o no la distancia de temps entre les extraccions de dades, i la de generar scripts sql amb l'estructura inicial, basant-se en les dades que obté de les fonts d'informació. Aquestes dues funcionalitats es descriuen més avall.

- ***crawlerFunctions.py***: Aquest és l'arxiu més gran, que conte totes les funcionalitats, inclosa la inserció en bloc, tot i que també conté la funció d'inserció antigua que insereix d'un en un. A continuació és detallen els noms de les versions de les funcions:

  - ***parseJsonAndInsert*** → Aquesta és la versió inicial que realitza insercions individuals. És la menys eficient i es recomana no utilitzar-la.

  - ***parseJsonAndInsert4Blocs*** → Aquesta versió és la millora de l'anterior, ja que agrupa les dades amb un mínim de 4 elements per grup (ja que inserim 4 dades diferents per estació). Es pot configurar el nombre d'insercions modificant la variable ***NUM_MESURES_PER_BAG*** que serà igual a ***TOTAL_PER_PETICIÓ*** = ***NUM_MESURES_PER_BAG*** *4.

  - ***parseJsonAndInsertBag*** → Aquesta és l'ultima versió i la més versàtil i optima. Permet realitzar insercions en bloc modificant la mateixa variable que anteriorment ***NUM_MESURES_PER_BAG***, però, aquest cop representa el nombre de dades per petició.

La variable ***CS_HTTP_ADDR*** definida al inici del fitxer representa la direcció de la API de commsensum a on inserirà la informació. Es recomana tenir el crawler a la mateixa màquina

per evitar tot el tràfic innecessari que es genera cada cop que s'insereixen dades, ja que en el nostre cas amb dos projectes es generen 2380 valors cada dos minuts.

Les variables ***CS_PROJECT_NAME***, ***CS_PROJECT_ID*** no són necessàries si s'utilitza la versió més optima ***parseJsonAndInsertBag***.

- ***config.py***: Aquest fitxer conte dades en format JSON, les quals són utilitzades pel crawler per saber a on buscar, quina informació extreure i a on inserir-la. Està dissenyat de manera que sigui fàcil inserir noves fonts d'informació. Per defecte hi ha dos configurades, la del Bicing de Barcelona i la de Saragossa. A continuació es detalla com afegir una nova font.

5.1.1. Afegir nou origen de dades

Per afegir una nova font d'informació s'ha de modificar el fitxer ***config.json***. A continuació es mostra l'exemple del Bicing:

```
{
"link":"http://wservice.viabicing.cat/v2/stations",
"name":"stations",
"nodeNamesJSON": {"address":[
        {"a1":"streetName",
        "a2":"streetNumber"}],
        "id":"id",
        "lat":"latitude",
        "lon":"longitude",
        "alt":"altitude"},
"staticNodeInfo":{"city":"Barcelona",
        "country":"SPAIN",
        "region":"Catalunya",
        "ccode":"BCN"},
"dataNamesJSON":{
        "status":"status",
        "total":{"c1":"bikes","c2":"slots"},
        "free":"slots",
        "notFree":"bikes"},
  "projectName":"urbanMobilityBCN",
  "userName":"gerard"
}
```

Primer de tot podem observar que hi ha diferents atributs amb diferents valors, on cada un te un significat específic. A la taula a continuació es detalla:

Nom	Significat
link	El link d'on es pot extreure el fitxer json amb les dades

name	El nom de l'atribut que conté l'array d'elements amb totes les dades
nodeNamesJSON	Conté un array amb els diferents noms dels atributs de cada dada, és a dir, el valor de "lat":"latitude" significa que en el origen de dades el valor de latitud s'escriu "latitude". El camp "address" pot contenir dues dades ja que pot tenir el nom del carrer i el numero per separat.
staticNodeInfo	Les dades de la font d'informació estàtica del projecte, és a dir, per exemple a informació d'origen de les dades del projecte.
dataNamesJSON	El nom dels atributs que contenen les dades que es recullen. El crawler només recull dades de l'estat de l'estació, quantes bicicletes en total hi ha, quantes lliures i quantes plenes.
projectName	Nom del projecte que voldràs utilitzar.
userName	L'usuari de la plataforma que ho inserirà

Taula 11: Detall fitxer configuració

Les dades com la API-Key per poder tenir permisos d'inserció s'ha de configurar manualment al inici del fitxer ***crawlerFunctions.py***.

Si s'afegeix una altre font s'ha de formatejar correctament en JSON incloent tots els atributs especificats a la taula anterior, sinó el crawler no funcionarà correctament.

Abans de començar a inserir dades amb el crawler fa falta crear l'estructura de dades dels projectes nous, dos fitxers per cada origen de dades. Per fer-ho fa falta executar el crawler de la següent manera:

python urbanCrawler.py sql

Amb aquesta comanda es generaran dos arxius per cada font d'informació que contindran les dades a inserir a la base de dades mysql, és a dir, la inserció del projecte, dels nodes i dels streams per cada projecte. En la primer font d'informació de Barcelona faria falta executar-los al servidor mysql amb les següents comandes:

mysql -u root -p < Barcelona-scriptNodes.sql
mysql -u root -p < Barcelona-scriptStreams.sql

Un cop finalitzada la inserció de l'estructura ja es pot configurar la seva execució periòdica al Cron, o bé executar-lo el crawler manualment (cada 2 minuts) amb la següent comanda:

python urbanCrawler.py 120

L'execució manual mostrà el temps que d'execució, i si introdueixes cntrl+c (ctrl+z) mostrarà la mitja de temps.

5.1.2. Requeriments

Per tal de poder utilitzar el crawler s'han d'instal·lar els següents components software:

sudo apt-get install python python-pip
pip install requests
pip install --upgrade pip

Tal com es pot observar només es requereix un mòdul de python, ***requests*** per poder realitzar les peticions HTTP a la API.

També fa falta tenir la API executada en algun servidor, tot i que lo recomanable es tenir-ho tot a la mateixa màquina per guanyar en velocitat d'execució i evitar tràfic innecessari.

5.1.3. Temps d'execució

Per tal de comprovar el rendiment en les diferents maquines de les que s'ha disposat, s'han realitzat un mínim de 10 execucions per cada tipus, calculant el temps mitjà en que la API absorbeix totes les peticions HTTP amb les dades i retorna una resposta al crawler. Aquest temps no vol dir el temps en el que les dades estaran actualitzades a la API, sinó el temps que tarda en executar-se el crawler. També s'ha de tenir en compte que algun cop, els servidors d'on s'obtenen les dades fallen i retornen error, per tant, el temps d'execució es veu una mica alterat.

Cal recalcar l'ús d'una Raspberry Pi B+ per fer la prova per veure si era factible o no tenir la API i el crawler funcionant en el dispositiu simultàniament. Els resultats són força interessants ja que aquest podria executar sense problemes totes dos aplicacions.

Una altre observació interessant és que com més ràpid és el hardware més rapida és l'execució en conjunt. Però, tal com és pot veure en la següent taula, a partir d'un cert hardware en condicions el temps d'execució no millora gaire, és a dir, que la diferencia de tenir un nucli a dos, o d'1GB de ram a 2GB no es gaire rellevant, almenys en la versió millorada del crawler i la API, en la que s'insereixen dades de 200 en 200.

	Cada 30 segons (seg.)	Cada 2 minuts (seg.)
RPI B+ (1CPU 512MB RAM)	**19.39**	**17.07**
VM Dep. AC (1CPU 1GB RAM)	**1.2**	**1.58**
VirtualBox VM (2CPU 2GB RAM)	**1.215**	**1.81**
VM (2CPU 8GB RAM)	**1.06**	**1.62**

Taula 12: Temps d'execució

A la taula a continuació està representada la diferencia d'utilitzar la versió inicial d'inserció individual, a la versió millorada que fa múltiples insercions (200 en aquest cas) per petició HTTP. L'execució s'ha programat cada 60 segons. En aquest cas si que importa més el

hardware, i com és pot veure la diferencia és rellevant. També podem veure que la Raspberry PI B+ ja no és prou rapida i es converteix en una opció inviable.

	Versió 1 (seg.)	Versió 2 (seg.)
RPI B+ (1CPU 512MB RAM)	**529.56**	**19.39**
VM Dep. AC (1CPU 1GB RAM)	**51.1**	**1.2**
VirtualBox VM (2CPU 2GB RAM)	**36.83**	**1.215**
VM (2CPU 8GB RAM)	**9.41**	**1.06**

Taula 13: Temps d'execució per versions

5.1.4. Possibles millores

Tot i que el codi és força modular, tenint el fitxer de configuració, el fitxer amb totes les funcions del crawler i el fitxer que executa tot el codi cada cert temps, cal generar més fitxers pensant en un futur proper. Per exemple quan s'implementin l'obtenció de dades d'altres formats com el XML, caldrà crear un altre fitxer addicional per aquesta funció. Per tant, reestructura el codi actual de manera que hi hagi un fitxer encarregat de les funcions pròpies del fitxer JSON i un altre per les funcions en general d'inserció de dades.

El fet de poder obtenir dades només en format JSON és una gran limitació. Una bona millora inicial seria implementar mòduls addicionals oferissin la funcionalitat d'obtenció de dades en formats XML i CSV. En una segona fase incorporar formats utilitzats en la web semàntica, com el RDF, per ampliar les capacitats del crawler.

L'execució actual és seqüencial i no s'aprofita el paral·lelisme. Una bona millora podria ser implementar mitjançant algun mòdul de Python, una versió que pogués executar-se en paral·lel i així aprofitat tots els nuclis del hardware que l'executa. Així podríem processar múltiples arxius amb informació simultàniament guanyant en temps d'execució.

La inserció de dades es fa de 200 en 200 elements per petició HTTP a la API. Aquest nombre no s'ha provat que fos el més adequat sinó que s'ha escollit com a nombre raonable. Una bona millora podria ser realitzar un script que analitzes quin nombre és el més adequat i que canvies dinàmicament segons la carrega de feina.

Totes les insercions es fan a través de la API, però no s'ha provat si es realitzessin directament a la BBDD ja que el crawler s'executarà en local en la mateixa màquina i podria tenir accés a ella. Sempre que es compleixi aquesta condició de que la BBDD i el crawler estiguin a la mateixa màquina, podria provar-se si és més efectiu inserir directament a la BBDD sense passar per la API.

5.2. Millores API REST

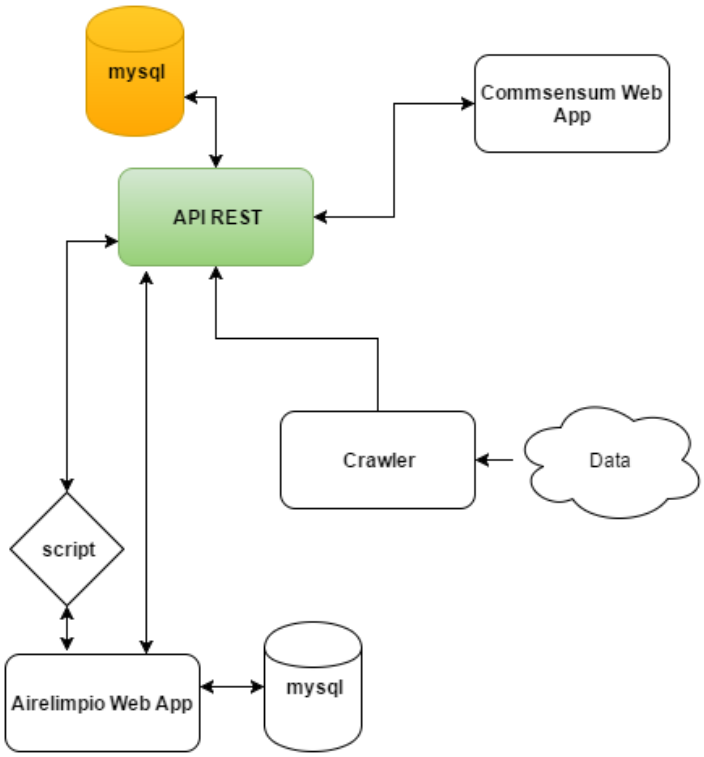
Actualment el projecte commsensum incorpora una API REST ja desenvolupada que s'encarrega de tota la inserció de dades dels diferents dispositius. Aquest mòdul està fet amb el llenguatge Node.js amb el framework Express, i utilitza la mateixa base de dades relacional que la plataforma web, és a dir, utilitza mysql. Així doncs, qualsevol usuari registrat a la plataforma amb permisos suficients, i en un projecte creat i preparat amb totes les dades necessàries, podria inserir informació utilitzant aquesta API fent crides HTTP.

La combinació d'aquestes dues tecnologies (node.js i mysql) és funcional per volums de dades petits menors a un centenar de gigabytes, i de moment la informació que emmagatzema és bastant menor. Però, en un futur, quan aquesta plataforma sigui publicada i l'utilitzi molta gent per enviar dades, farà falta una tecnologia d'emmagatzematge més escalable, és a dir, una base de dades noSQL.

Aquest mòdul també te altres mancances, per exemple, en la versió actual no es poden inserir varies dades en una sola petició. Això es pot convertir en un problema bastant greu si es volen fer moltes insercions seguides en un mateix moment. Ja que el que jo he estat desenvolupant és un crawler que fa justament grans nombres d'insercions en un mateix instant, m'he trobat que el mòdul es satura perquè li arriben moltes peticions HTTP simultànies (2380 cada dos minuts des del meu crawler), i hi ha molt overhead per mantenir totes aquestes connexions. El crawler només recull dades de dos fonts (de Barcelona i Saragossa), però quan hi hagi un crawler real que insereixi informació des de moltes més fonts, la API segurament tingui problemes per servir totes les peticions.

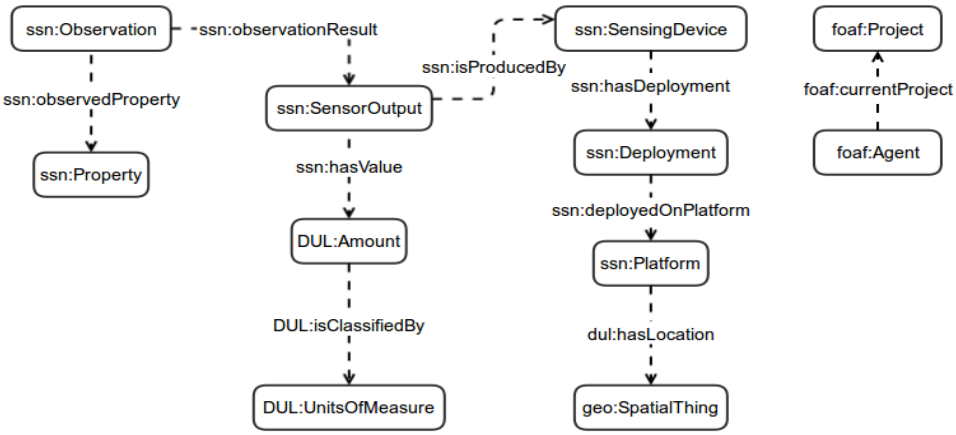
Després de trobar els problemes descrits, s'ha decidit implementar la primera millora, i plantejar la segona. La primera permetrà a la API rebre un paquet amb múltiples dades a inserir. La segona, com a primera fase permetrà inserir les dades també a una base de dades noSQL, a MongoDB. S'ha triat aquesta perquè és molt compatible amb Node.JS i Express, a part de tenir un molt bon rendiment, escalabilitat, i permet guardar la informació directament en format document, és a dir, JSONs en el nostre cas. Les primera millora esta implementades al repositori de codi bitbucket al projecte general, i la segona ja que esta en proves, en el servidor de proves.

A continuació es pot observar l'arquitectura, on es pot veure que la API REST treballa amb una base de dades mysql. Aquesta rep peticions de de la web de commsensum, del script que executa la aplicació Airelimpio, directament des de Airelimpio i des del crawler.



Imatge 9: API REST

També feia falta analitzar si la definició ja implementada de la ontologia servia pel nou contingut. El que es vol representar són dades amb una certa unitat, generades per una estació amb una localització, dades que estan associades a un projecte realitzat per una persona. Tots aquests conceptes estan representats amb la definició actual per lo que s'ha decidit mantenir el disseny actual, per tant, les dades d'aquest projecte es compartiran amb la mateixa ontologia. A la següent imatge és pot veure amb més detall:



Imatge 10: Model ontològic

5.2.1. Inserció en bloc

Per tal de realitzar unes insercions ràpides ha sigut implementada una millora a la API que permet enviar fins a 200 dades en una sola petició HTTP. Aquesta nova funcionalitat la utilitza el crawler i li permet inserir tota la informació molt ràpidament.

La petició HTTP individual es fa a una direcció que es genera a partir del nom del projecte i del nom del stream a on s'inserirà la informació, afegint les dades al body. A continuació hi ha un exemple:

http://commsensum.com:3000/v1/projecteA/Stream-2-N02
header {dades autenticació}
body {valor:'3',%,date:'now'}

A la nova petició dissenyada per inserir múltiples dades ja no li fa falta incorporar el nom del stream a la direcció HTTP, sinó que aquesta informació és inserida a dintre d'un objecte JSON al body. A continuació un exemple:

http://commsensum.com:3000/bag/v3/projecteA
header {dades autenticació}
body {[{stream:'Stream-2-No2',valor:'3',%,date:'now'},{stream:'Stream-3-No2',valor:'3',%,date:'now'}]}

Com es pot observar, s'enviarien dos elements en una mateixa petició que anirien dintre del mateix projecte, però en diferents streams. El límit de dades per petició està configurat a la API i pot ser canviat. De moment s'ha decidit establir-lo a 200 elements. A l'annexa A hi ha documentació tècnica més detallada.

5.2.2. Inserció a MongoDB

La limitació de les bases de dades relacionals m'ha obligat a plantejar-me la utilització d'una base de dades més escalable, una noSQL, en concret MongoDB. Una gran avantatge rellevant respecte les bases de dades relacionals és que aquesta pot escalar horitzontalment, és a dir, si augmenta la quantitat de dades, afegint més nodes s'aconsegueix mantenir el rendiment. Si pensem en els grans volums de dades que haurem de gestionar en un futur proper, les relacionals no estan preparades. Una altre millora important de MongoDB és que aquesta permet realitzar anàlisis de les dades de qualsevol tipus directament a la base de dades i retornar resultats quasi en temps real. Això permetria en certs casos estalviar-nos la utilització d'un altre clúster Big data (Hadoop per exemple) que analitzes les dades. Finalment també podem veure com avantatge el disseny schema-less que ofereix mongoDB, permetent inserir dades de diferents formats i/o columnes, sense estar limitat pel disseny de les taules.

Per falta de temps i perquè no és un objectiu principal del projecte, només s'ha experimentat amb les insercions simultànies. A demes, com que utilitzar-la per complet com a substitut de la

base de dades mysql implicaria re-dissenyar tota la plataforma, incloent la API i la plataforma web, s'ha decidit en aquesta primera fase inserir les dades addicionalment a mongoDB. Es provarà si aquesta nova funcionalitat afegeix algun overhead i si les dades estaran estabilitzades abans. En alguns entorns de proves s'ha observat que si s'executa el crawler cada dos minuts, al cap d'un temps, les dades de la base de dades no estan actualitzades, i s'ha tingut que esperar uns 10 minuts . Aquest problema és manifesta en sistemes amb hardware no gaire potent, i sembla més un problema de mysql d'escriptura en disc que no pas de la API. Més en concret, ha passat en l'execució a una RPI B+ i a una màquina virtual de proves del departament d'AC que només tenia 1 nucli i 1 GB de ram, quan l'execució es realitzava 30 segons, un temps molt curt.

Pel que fa la implementació, s'ha utilitzat un mòdul de Node.js anomenat mongodb que permet realitzar qualsevol tipus d'operació sobre mongoDB. Prèviament s'ha configurat un servidor mongod estàndard per realitzar les proves en la mateixa màquina. A l'annexa B hi ha informació tècnica detallada.

5.2.3. Possibles millores

Tot i que s'ha pogut provar la inserció de dades a mongoDB, no s'ha fet la implementació de consulta, obtenint un resultat que no tenia cap sentit, inseria dades però després no les consultava. La millora que es proposa és incorporar mongoDB enlloc de mySQL per emmagatzemar totes les dades dels sensors. Això implicaria, configurar el servidor mongoDB, utilitzar algun mòdul per treballar amb mongoDB per Node.js i modificar totes les funcions actuals que treballen amb la taula Data de mysql. També caldria investigar si des d'algun altre lloc es consulten les dades d'aquesta taula com per exemple des de l'aplicació web.

També és podria utilitzar mongoDB com a substitut complert de mysql, però implicaria modificar totalment la API, i també la aplicació web. Per tant, aquesta millora requereix moltes hores, i és convenient planificar-ho. Les millores en quant a velocitat i escalabilitat és notarien a llarg termini, però s'hauria de configurar el servidor mongoDB com a un clúster, inicialment de dos nodes.

Caldria fer una auditoria de seguretat sobre l'aplicació, per veure si almenys te les mínimes mesures de seguretat. Cada cop està més a prop la seva versió final i és important que no tingui cap forat de seguretat que s'alliberi el codi font. També s'ha de tenir en compte que com més gran es faci l'aplicació més difícil serà d'analitzar si és més o menys segura, per tant millor fer aquest examen quan abans millor.

Les funciones que disposa són suficients però per millorar la seva qualitat caldria afegir-li més funcionalitats. Per exemple poder consultar els projectes públics que existeixen a la plataforma amb una sola crida, obtenir les ultimes 10 dades d'un cert projecte públic, poder escollir el format en el qual es volen les dades, ja sigui utilitzant el format de web semàntica o bé un simple JSON, etc. Això implicarà afegir més funcions publiques, que si s'utilitzen normalment faran augmentar la carrega de treball de la aplicació.

Cada cop és més normal utilitzar clústers, és a dir, conjunt de dos o més maquines que funcionen com a una. Així doncs una bona millora podria ser distribuir la feina entre varies maquines. Node.js te un mòdul que s'anomena Clúster que permet fer-ho sense la necessitat de tenir diferents maquines. Podria ser una primera solució, tot i que estaríem confiant 100% en aquest mòdul, i si hi hagués algun problema hauríem de tornar a la versió anterior. Una altre alternativa seria tenir dos o més maquines amb l'aplicació, i un balancejador de carrega que s'encarregaria de rebre totes les peticions i distribuir-les entre els diferents nodes. Implicaria tenir dos o més maquines replicades i una addicional que faria la funció de balancejador.

5.3. Aplicació web

Per a realitzar l'aplicació web s'ha partit de base del projecte Airelimpio desenvolupat per una estudiant de Màster. Utilitza la base de dades relacional mysql i està feta amb Django, un framework de Python. Aquesta aplicació no ha estat configurada per a que funcioni en producció, i no està del tot optimitzada. La versió base permet mostrar en un mapa les dades d'un projecte de commsensum que conté les dades dels sensors de Catalunya i Madrid que prenen mides de la qualitat de l'aire. Per actualitzar les dades utilitza un script en Python que s'executa amb el Cron, que el que fa és consultar si les dades estan actualitzades o no fent una petició a la API de commsensum.

Les dades es mostren en un mapa que utilitza la API en Javascript de google maps. Per tant, ha calgut actualitzar la clau per tal de poder seguir visualitzant el mapa. Els nodes es representen amb un icona que segons l'estat de l'estació, es mostra d'un color o d'un altre. Per exemple si la estació fa dies que no s'actualitza, s'utilitza un icona gris. Si la estació te valors correctes per sota dels límits, mostra un icona verd. Si pel contrari algun valor està per sobre del límit es mostra la icona vermella. Per tal de facilitar el desenvolupament s'ha utilitzat la icona taronja per representar les estacions de bicicletes.

Les funcionalitats mínimes que te que disposar la nova versió, per començar, te que mostrar les estacions de bicicleta al mapa amb un icona que sigui distintiu. Després per cada icona al fer clic sobre haurà de mostrar les dades més recents d'aquella estació. Com a últim requeriment, la aplicació s'haurà d'anar actualitzant periòdicament. Una gran limitació és el temps, ja que s'ha planejat dedicar-li només 10 hores, perquè es considera un objectiu secundari.

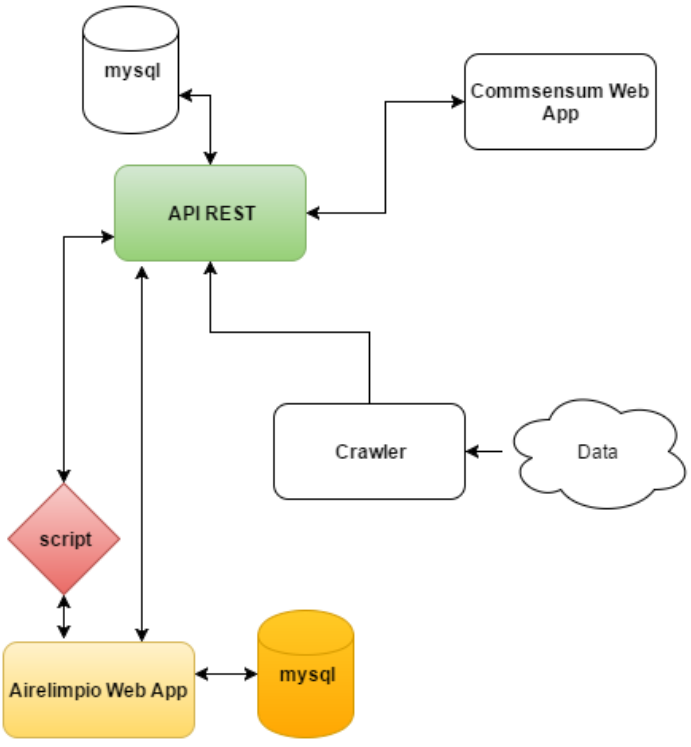
Per tal de complir amb aquestes tres funcionalitats especificades en el paràgraf anterior s'ha tingut que analitzar el codi font i fer els següents canvis:

views.py→ Aquest fitxer que és utilitzat pel framework de Django per gestionar la lògica de les vistes de la pàgina web. Aquest és el que consulta les dades de la base de dades local mysql i envia les necessàries perquè es renderitzin al HTML final. En la versió base només tenia en compte les dades del projecte pel qual va ser fet, per tant, s'han afegit els dos altres projectes

de Commsensum. Per afegir-los s'ha tingut que modificar la consulta SQL que obtenia la informació. En aquest fitxer també s'ha canviat que totes les estacions d'aquests dos altres projectes utilitzin un icona taronja per defecte i així poder diferenciar-les.

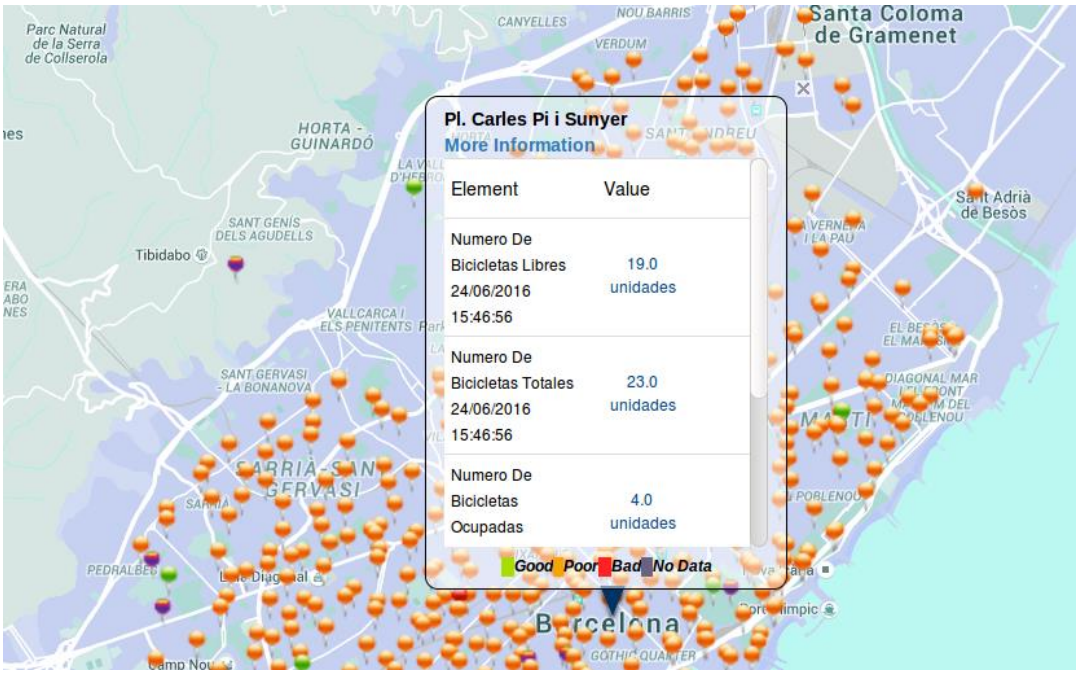
tareaMonitoreo.py → Aquest fitxer és el que serveix per actualitzar les dades de la pàgina web airelimpio. A la versió original, cada cert temps es va executant a través del Cron. La versió base tant sols crida a la funció principal que te ja configurat el projecte i totes les dades necessàries. Així doncs, s'ha canviat la funció principal per una que permet especificar el numero de projecte i l'origen de les dades. Això ens permet actualitzar les dades de múltiples projectes i fonts cridant l'script un cop. Cal també canviar la funció conexion() amb les credencials corresponents a la base de dades mysql local. Els identificadors dels projectes pertinents de Commsensum de Barcelona i Saragossa afegits han de concordar. Aquest script al executar-se tarda un temps considerable i per tant es descarta posar-lo en producció a la versió final. Es important pensar en una millora d'aquest procés perquè la actual no es viable quan s'ha d'actualitzar molta informació de moltes estacions.

A continuació es pot observar l'arquitectura de l'aplicació, que funciona amb una base de dades mysql, i que executa periòdicament un script en Python que recopila dades de la API REST. Addicionalment també es fan peticions directes des de l'aplicació per obtenir rangs de dades.



Imatge 11: Aplicació web

A la imatge següent es pot veure la versió final de la interfície, on es pot observar que sobre el mapa de google maps hi ha etiquetes, on les de color ataronjat representen les estacions de bicicletes. Quan es clica sobre una d'elles es mostra el detalls dels quatre valors que conté, conjuntament amb la data de quan han sigut actualitzades.



Imatge 12: Interfície web

A l'annexa C es detalla la posada en marxa de l'aplicació per el desenvolupament.

5.3.1. Possibles millores

La versió actual permet visualitzar les dades més recents al mapa, però no te implementada la actualització automàtica, perquè per falta de temps no s'ha desenvolupat. Aquesta millora és imprescindible per tenir les dades en temps real.

El script que realitza la actualització de les dades no és gaire eficient i tarda massa temps, i això que només hi ha tres projectes. Això és un problema si es vol tenir les dades en temps real. Caldria optimitzar aquest procés al màxim perquè sense les dades actualitzades no te gaire sentit l'aplicació.

La interfície gràfica és acceptable i permeti fer les accions bàsiques com mostrar els sensors en un mapa, i quan cliques sobre d'ells mostra les diferents dades. També permet filtrar per tipus de sensor. Però, per exemple seria convenient utilitzar icones diferents més representatius, enlloc dels icones simples que s'utilitzen. Una altre millora podria ser agrupar els sensors a la vista, perquè no es mostrin millers d'estacions a la vegada, que faci que sigui a vegades illegible i que vagi més lenta l'aplicació.

La utilització d'una base de dades relacional és funcional quan no hi ha molta informació, com passa actualment que només hi ha tres projectes. Però, quan hi hagi molta més informació caldrà replantejar l'ús d'una base de dades noSQL, que ens ajudarà a mantenir el rendiment, ja que estan dissenyades per gestionar grans volums de dades i escalar horitzontalment.

També és podria plantejar alguna tècnica de caching de les dades que es demanen a la API, és a dir, cada cop que es demanen les dades a la API guardar-les localment temporalment per si algun altre usuari les torna a demanar, les mateixes o subconjunts de les mateixes. Hi ha mòduls de Node.js que ho permeten implementar, tot i que també es podria realitzar amb alguna base de dades.

6. Anàlisis

En el següent apartat s'analitza els tipus de llicencia que més s'utilitzen, i quines estan definides a les dades que treballem

6.1. Llicencies en Open Data

És molt important abans d'emmagatzemar dades comprovar quin tipus de llicencia té, ja que podríem estar infringint la llei. A continuació es detalla la més utilitzada:


Per tal de controlar els drets d'autor es van crear les Creative Common, les quals permeten a l'autor de l'obra definir més en detall quins són els drets sobre les obres en concret. La última versió és la 4.0, tot i que es poden seguir utilitzant versions antigues.

Aquestes poden definir-se utilitzant combinacions de les següents quatre condicions generals:


Atribució (BY): El beneficiari te dret de copiar, distribuir, exhibir i representar l'obra i fer obres derivades sempre i quan reconegui i citi la obra de la forma especificada per l'autor.


No comercial (NC): El beneficiari de la llicencia te el dret de copiar, distribuir, exhibir i representar la obra i fer obres derivades per fins no comercials.


No derivades (ND): El beneficiari de la llicencia només te dret de copiar, distribuir, exhibir i representar copies literals de la obra i no te dret de produir obres derivades.


Compartir igual (SA): El beneficiari de la llicencia te el dret de distribuir obres derivades sota una llicencia idèntica a la llicencia que regula la obra original.

Així doncs existeixen sis possibles combinacions llicencies, ja que la BY és obligatòria, SA i ND són incompatibles, i NC és opcional.

6.1.1. Llicencia Bicing

Les dades publiques des del 09/11/15 al portal d'open data de Barcelona tenen definida la llicencia CC BY 3.0[25], el que significa que tenim dret a copiar, distribuir, exhibir, representar l'obra i fer obres derivades, sempre i quan reconeguem i citem l'autor de forma especifica. Per tant, podem utilitzar i emmagatzemar les dades sense problemes.

6.1.2. Llicencia Bicing Saragossa

Les dades de les estacions de bicicletes de Saragossa estan obertes des del 10/03/15 i no utilitzen la nomenclatura CC, sinó que utilitzen una llicencia personalitzada amb condicions[26]. Aquesta permet la seva reutilització sempre i quan es compleixin les següents condicions d'ús:

- Esta prohibit desnaturalitzar el sentit de la informació
- Es obligatori citar la font del objecte a reutilitzar. Es podria fer mitjançant el text "Origen de les dades: Ajuntament de Saragossa".
- Es obligatori mencionar la data de la ultima actualització del objecte reutilitzat, sempre i quan estigues inclosa al document original.
- No es podrà indicar, insinuar o suggerir que l'Ajuntament de Saragossa participa, patrocina o suporta la reutilització de les dades.
- Els meta-dades amb les dades de actualització i les condicions de reutilització no s'han de alterar, sinó que s'han conservar.

Per tant, com que complirem les condicions, podem utilitzar les dades sense problema.

Per més informació consultar el enllaç de la bibliografia a l'apartat 2.3.3.

6.2. Escalabilitat

El gran volum de dades m'ha fet plantejar-me les necessitats d'escalabilitat en quant a processament de les dades. S'ha de tenir en compte que el creixement de dades cada cop és més elevat i tard o d'hora farà falta redissenyar-ho tot sinó s'ha pensat prèviament en l'escalabilitat.

6.2.1. Anàlisi d'informació

A la actual base de dades mySQL està definida una taula anomenada Data, on es guarden totes les dades que s'insereixen. Segons les especificacions consta dels següents camps, que s'han fet servir per analitzar el volum de dades que generarien la inserció de les dades del crawler. Cal recalcar que es fan 2380 insercions cada 2 minuts, enviant l'estat de totes les estacions de bicicletes de Barcelona i Saragossa, és a dir, un gran volum de dades que probablement creixi en un futur proper. A continuació es detallen les mides de les dades de la taula Data:

Nom	Tipus	Mida
id	int	4 Bytes
idStream	int	4 Bytes
Value	varchar(450)	4 Bytes *
Date	datetime	8 Bytes
idDataType	int	4 Bytes
Total per fila	24 Bytes	
Índex	Total per fila * 2 = 48 Bytes	
Total	**72 Bytes**	

Taula 14: Detall inserció

*~3 Bytes de mitja + 1 Byte (si requereix <256 Bytes) = 4 Bytes

Realitzant una inserció de dades s'ha calculat la diferencia de la mida de la taula, i dona valors lleugerament superiors. Cal recalcar que els resultats són orientatius.

Temps	Càlcul (total*insercions*n)	Quantitat
Diari	Total * 2380 * 30 * 24	123,38 MB/dia
Mensual	TotalDiari * 30	3,7 GB/mes
Anual	TotalDiari * 365	45,03 GB/any

Taula 15: Quantitat per interval

Un cop ja estimat el volum de dades que caldrà guardar, podríem encara dir que no comptem amb quantitats descontrolades de dades, i que no faria falta replantejar el tipus de base de dades que s'utilitza. Però, si multipliquem aquest volum de dades per tres (11,1 GB per mes), és a dir, si obtinguéssim informació de sis ciutats, llavors ja començaríem a tenir una base de dades força gran, i hauríem de començar a pensar en característiques com la escalabilitat. També observaríem que el rendiment en general començaria a decaure, ja que no és el mateix buscar dades entre 10 GB que entre 100 GB.

6.2.2. Bases de dades noSQL

També anomenat "no només SQL", són un tipus de bases de dades que cada cop s'utilitza més. Una de les principals característiques a diferencia dels models de bases de dades relacions és que aquest normalment no compleix els quatre atributs ACID (Atomicitat, Consistència, Aïllament i Durabilitat). Una altre característica important és que aquestes estan dissenyades per a poder escalar en horitzontal sense problemes.

En general, aquestes estan orientades a solucionar problemes en entorns Big Data o el Internet de les coses, on es mouen grans volums de dades. En aquests entorns apareixen les tres V:

- ***Velocitat***: La velocitat en que es genera informació és molt elevada
- ***Varietat***: Hi ha informació molt variada, de molts tipus i formats
- ***Volum***: El volum de dades és molt gran.

Si almenys dos d'aquestes tres característiques es manifesten, cal replantejar si el sistema gestor de base de dades és l'adequat, i lo més probable és que la millor solució sigui utilitzar una de tipus noSQL.

Hi ha cinc subtipus que les classifiquen segons la seva estructura:

- **Columna**: La informació enlloc de guardar-se en files, es guarda en columnes. Aquest canvi tot i que pugui semblar que podria empitjorar el rendiment, permet tenir un gran rendiment i escalabilitat. Un exemple és HBase, una base de dades que ve incorporada amb Hadoop.
- **Document**: La informació que conte són documents, és a dir, dades encapsulades en estàndards com XML, JSON, o BSON. Normalment aquests tipus ofereixen una API que permet buscar les dades tenint en compte el contingut de les dades. Un exemple és MongoDB, que enlloc de taules te collections, i enlloc de files te records.
- **Clau-valor**: La informació es guarda en una espècie de diccionari, on s'associa una clau amb una dada. La dada normalment pot ser de quasi qualsevol tipus, depenent de la base de dades. Un exemple és Redis o Amazon DynamoDB.
- **Graf**: La informació es guarda en format de graf, és a dir, consta d'elements interconnectats amb relacions. Aquest tipus va força bé per a analitzar la informació aplicant algorismes de la teoria de grafs. Un exemple és Neo4j.
- **Multi-model**: La informació es guarda en diferents formats, és a dir, combina les característiques de dos o mes tipus dels detallats anteriorment. Un exemple és ArangoDB.

En general totes les bases de dades d'aquests tipus tenen les següents característiques:

- **Descentralització**: Normalment les bases de dades són distribuïdes i descentralitzades en un clúster, és a dir, cada node te la mateixa importància que els demes. Per tant, si un node falles, la base de dades seguiria funcionant.

- **Escalabilitat**: Totes elles han estat dissenyades perquè puguin escalar horitzontalment, és a dir, afegint més nodes al clúster distribuït, mantenir el rendiment sense problemes. Les bases de dades relacionals pateixen d'aquest problema i no poden escalar tant fàcilment sense perdre rendiment.

- **Econòmic**: En molts casos on hi ha molt volum de dades és més econòmic utilitzar un clúster amb una base de dades noSQL, ja que aquest normalment no requereix un hardware high-end, a diferencia de les bases de dades relacionals, i ofereix més qualitat/preu.

- **Replicació**: Per tal de lidiar amb les possibles fallades d'algun node, apareix la replicació d'informació. Cada bloc d'informació es replica entre altres diferents nodes del mateix clúster permetent no només evitar perdre la informació, sinó també millorar en rendiment.

Després de buscar informació sobre aquests tipus de base de dades, s'ha valorat la implementació de tres d'elles, una de cada tipus, totes elles gratuïtes:

- **MongoDB**: De tipus document, creada el 2009, molt utilitzada a l'actualitat, amb gran compatibilitat amb Node.js i Express. Permet fer consultes complexes. Utilitzada per Facebook, Bosch, Expedia, etc.
- **Cassandra**(tipus columna-ample): És de tipus columna i és un projecte d'Apache iniciat el 2008, molt ràpida, i surt molt ben parada en alguns tests de rendiment. L'utilitzen al CERN, a Apple, a Netflix, etc.

- **ArangoDB**(tipus multi-model): És de tipus multi-model, i és la més recent de totes, no és gaire coneguda, creada el 2011 i també propietat d'Apache. Permet ser utilitzada com a model Clau-Valor, Document i Graf, la que la converteix en la més versàtil.

Però, per falta de temps, s'ha decidit implementar només la primera, MongoDB, per obrir les portes a noves possibles millores sobre la plataforma.

6.2.3. Disseny base de dades

MongoDB és Schema-free, és a dir, és independent del disseny, no fa falta crear cap disseny previ. En el meu cas com que només s'inseriran dades d'estacions de bici, només farà falta crear una base de dades i una taula (collection a MongoDB). En aquesta taula s'inseriran totes les dades en format JSON a la vegada que s'insereixen a mysql.

A la vegada també s'haurà de modificar la consulta de dades, perquè enlloc de buscar a la base de dades mysql, busqui a la nova noSQL. No te cap sentit només realitzar les insercions ja que no hi hauria cap millora, tot el contrari, aniria més lenta tota l'aplicació web per culpa d'aquesta millora en la API. Per tant, per a realitzar tots els canvis necessaris s'ha estimat un nombre considerable d'hores, i com a conseqüència s'ha decidit cancel·lar la implementació per falta de temps.

No obstant, el plantejament de com hauria de fer-se la millora i alguna prova si que s'ha pogut fer. Tal com s'ha comentat a apartats anteriors, la taula Data és la taula que conte més informació, per tant és la que hauria de ser implementada en mongoDB. No hi ha necessitat d'utilitzar la nova base de dades per tot, ja que la relacional fa bé la seva feina en volums d'informació petits i relacionats que no varien gaire. Per tant, tant sols s'hauria de modificar les insercions i consultes a aquesta taula, substituint les consultes mysql per les funcions pertinents de mongoDB. Aquest canvi tard o d'hora farà falta per poder treballar amb grans volums de dades amb temps de resposta acceptables. També permetrà poder utilitzar un clúster enlloc d'un sol servidor mongoDB, guanyant en seguretat de les dades i en rendiment.

7. Conclusions finals

La problemàtica plantejada inicialment conjuntament amb els objectius definits han sigut solucionats a temps, tal com estava planificat. Com a resultat s'ha realitzat un crawler que recopila les dades del Bicing de Barcelona i de Saragossa, i inserta les dades a la API REST modificada eficientment en bloc cada dos minuts. Seguidament hi ha un procés que s'executa a la pàgina web que agafa les dades tant de pol·lució com de les estacions de bicicletes i les inserta a la pàgina web modificada, la qual mostra un mapa amb les dades actualitzades. Les dades noves inserides a la API REST estan publiques per les persones autoritzades en format RDF/XML.

S'han tingut en compte els obstacles detectats trobant alguna solució adequada. Per exemple, el problema d'inserir 2380 dades de cop cada dos minuts, s'ha resolt modificant la API REST perquè acceptes peticions amb múltiples dades, i així s'ha pogut millorar molt en eficiència. Per al problema de poder afegir múltiples orígens de dades, s'ha separat un fitxer de configuració en format JSON on s'especifiquen totes les dades necessàries per efectuar la inserció. En general tots els impediments que s'han anat trobat durant la realització del treball de fi de grau s'han pogut resoldre sense problemes, millorant aquesta habilitat necessària com a futur enginyer.

La sostenibilitat era un dels aspectes que es pretenia treballar amb aquest projecte, ja que clarament els objectius personals eren millorar la vida de les persones, facilitant més informació d'interès per elles com és la qualitat de l'aire i l'estat de les estacions de bicicleta en una mateixa aplicació. Els usuaris que normalment utilitzen el servei de bicicletes i que estan preocupats pel medi ambient són els més beneficiats. I no només això, sinó que aquest projecte obre dos grans portes. La primera és la possibilitat d'incorporar més tipus de dades a la plataforma i obtenir valor d'aquestes, per exemple dades del transport públic o del nivell de soroll. La segona és treure-li més utilitat a la informació que ja es disposa, com per exemple crear una aplicació que amb aquestes dues dades et mostri la millor ruta per anar en bici d'un origen a un destí ja que els ciclistes passen molt de temps rodejats de vehicles emissors de grans quantitats de partícules contaminants.

Segons el meu punt de vista s'haurien de fer més projectes d'aquest estil a cada semestre, ja que considero que ha sigut a part d'interessant, molt enriquidor. Amb cada projecte et trobes amb reptes diferents, noves tecnologies, nous companys de grup, pots esbrinar què és el que realment t'agrada, què és el que se't dona millor, etc. En el meu cas he pogut descobrir que tinc interès en projectes enfocats en el medi ambient, i amb el objectiu de millorar la vida de les persones. De fet m'ha agradat tant que m'agradaria continuar treballant en aquest projecte, i crec que aquesta és la sensació amb la que t'has de quedar, amb ganes de millorar i d'aprendre.

Annexa A

Preparar l'entorn de treball:

Primer de tot actualitzar el software de la màquina:

```
sudo apt-get update
apt-get upgrade
```

Instal·lar els paquets necessaris:

```
sudo apt-get install mysql-server libmysqlclient20 npm git
```

Clonar el repositori a una carpeta:

```
git clone https://kulsur@bitbucket.org/commsensum/general.git
```

Anar a la carpeta de la api, i Instal·lar el paquet necessaris de nodejs com a global (per instal·lar node posteriorment)

```
sudo npm install n --global
sudo npm install npm@3.8.8 node-gyp@3.3.1 -g
npm install
sudo n@5.8
```

Copiar el backup de la bbdd mysql amb la comanda scp i executar la següent comanda per restaurar:

```
mysql -u root -p < backup_bbdd.sql
```

Executar la API

```
node commsensum.js
```

Pel que fa el crawler, primer instal·lar dependències:

```
sudo apt-get install python python-pip
pip install requests
```

```
pip install --upgrade pip
```

Descarregar els fitxers urbanCrawler.py, crawlerFunctions.py i config.json del crawler amb scp o similar.

Executar el crawler per generar els scripts sql inicials:

```
python urbanCrawler.py sql
```

Inserir les dades generades a la bbdd mysql local:

```
mysql -u root -p < Barcelona-scriptNodes.sql
```

```
mysql -u root -p < Barcelona-scriptStreams.sql
```

```
mysql -u root -p < Zaragoza-scriptNodes.sql
```

```
mysql -u root -p < Zaragoza-scriptStreams.sql
```

Executar el crawler per anar inserint dades cada 2 minuts:

```
python urbanCrawler.py 120
```

Annexa B

Instal·lar mongodb server i client:

```
sudo apt-get install mongo-server
```

Inicia client amb dades per defecte, executar consolar i comprovar que funciona amb la comanda següent:

```
mongo
```

Crea la base de dades:

```
use test
```

Crear taula:

```
db.createCollection('taula')
```

Inserir dada, on cada element és únic perquè es genera un Object id automàtic:

```
db.taula.insert({name:'pepe'})
db.taula.insert({name:'pepe',address:'lolailo'})
```

Buscar element:

```
db.taula.find({name:'pepe'})
```

Comandes útils d'ajuda:

```
db.help() o db.mycoll.help()
```

Dintre de la carpeta de la api instal·lar el paquet de nodejs:

```
npm install mongodb
```

Executar el crawler ja preparat en el apartat anterior per inserir les dades:

```
python urbanCrawler.py 120
```

Annexa C

Els següents passos s'han realitzat per tal de poder desenvolupar la aplicació Airelimpio.

Primer de tot restaurar la base de dades inicial, que està a un repositori:

git clone https://kulsur@bitbucket.org/commsensum/backup-bd-airelimpio.git

cd ./backup-bd-airelimpio/daily/monitoreo

gzip -d monitoreo_2016-03-09_12h12m.Wednesday.sql.gz

mysql -u root < monitoreo_2016-03-09_12h12m.Wednesday.sql

Instal·lar dependències:

sudo apt-get install python python-pip python-dev libmysqlclient-dev libcurl4-openssl-dev

sudo apt-get install build-essential autoconf libtool pkg-config supervisor nginx gunicorn

Copiar el codi font del repositori:

git clone https://kulsur@bitbucket.org/commsensum/backup-airelimpio.git

Instal·lar les dependències de Python amb el fitxer requirements.txt:

sudo pip install -r requirements.txt

Editar fitxer de configuració, per canviar la ruta als fitxers estàtics per exemple:

vi /monitoreoCA/settings.py

Bibliografia

[1] Compnet research group, « Commsensum platform, » [En línia]. Disponible a: http://compnet.ac.upc.edu/intranet/?q=node/199 [Últim accés: 02/03/2016]

[2] Ajuntament de Barcelona, « OpenDataBCN, » [En línia]. Disponible a: http://opendata.bcn.cat/opendata/ [Últim accés: 02/03/2016]

[3] Aplicació AireLimpioYa, « AireLimpioYa, » [En línia]. Disponible a: http://airelimpio.pc.ac.upc.edu/ [Últim accés: 02/03/2016]

[4] Thingspeak platform company, « Thingspeak main page, » [En línia]. Disponible a: https://thingspeak.com [Últim accés: 02/03/2016]

[5] TheThings.io platform company, « TheThing.io main page, » [En línia]. Disponible a: http://thethings.io [Últim accés: 02/03/2016]

[6] Wikipedia, « Semantic web, » [En línia]. Disponible a: https://en.wikipedia.org/wiki/Semantic_Web [Últim accés: 31/03/2016]

[7] W3C, « OWL, » [En línia]. Disponible a: https://www.w3.org/2001/sw/wiki/OWL [Últim accés: 31/03/2016]

[8] W3C, « RDF, » [En línia]. Disponible a: https://www.w3.org/2001/sw/wiki/RDF [Últim accés: 31/03/2016]

[9] W3C, « RDFS, » [En línia]. Disponible a: https://www.w3.org/2001/sw/wiki/RDFS [Últim accés: 31/03/2016]

[10] Wikipedia, « JSON, » [En línia]. Disponible a: https://en.wikipedia.org/wiki/JSON [Últim accés: 31/03/2016]

[11] Wikipedia, « HTTP, » [En línia]. Disponible a: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol [Últim accés: 31/03/2016]

[12] Wikipedia, « HTML, » [En línia]. Disponible a: https://en.wikipedia.org/wiki/HTML [Últim accés: 31/03/2016]

[13] Wikipedia, « Bootstrap framework, » [En línia]. Disponible a: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) [Últim accés: 31/03/2016]

[14] Djangoproject, « Main webpage, » [En línia]. Disponible a: https://www.djangoproject.com/ [Últim accés: 31/03/2016]

[15] Nodejs, « Main webpage, » [En línia]. Disponible a: https://nodejs.org/en/ [Últim accés: 31/03/2016]

[16] Michael Page company, « Salarios especialistes tecnologia, » [En línia]. Disponible a: http://www.michaelpage.es/sites/michaelpage.es/files/tecnologia2015.pdf [Últim accés: 12/03/2016]

[17] OVH company, « Tarifas Servicios servidores dedicados, » [En línia]. Disponible a: https://www.ovh.es/servidores_dedicados/ [Últim accés: 12/03/2016]

[18] Lorente y Lorente, « Amortizaciones, » [En línia]. Disponible a: http://www.lorenteylorente.es/2015/03/como-aplicar-las-nuevas-tablas-de-amortizacion-en-2015/ [Últim accés: 12/03/2016]

[19] Microsoft company, « Products prices, » [En línia]. Disponible a: http://www.microsoftstore.com/ [Últim accés: 12/03/2016]

[20] Endesa, « Tarifas, » [En línia]. Disponible a: http://www.endesaone.com/one-luz [Últim accés: 12/03/2016]

[21] Aigües de Barcelona, « Tarifes, » [En línia]. Disponible a: http://www.aiguesdebarcelona.cat/facturadelaigua [Últim accés: 12/03/2016]

[22] Telefonica-Movistar, « Tarifas, » [En línia]. Disponible a: www.movistar.es [Últim accés: 12/03/2016]

[23] Tu huella ecológica, « Calculo energía, » [En línia]. Disponible a: http://www.tuhuellaecologica.org/encuestas/energia.asp [Últim accés: 18/06/2016]

[24] Cámara de Zaragoza, « Calculador de emisiones, » [En línia]. Disponible a: http://www.camarazaragoza.com/wp-content/uploads/2012/10/calculoemisiones.xls [Últim accés: 18/06/2016]

[25] Open data Barcelona , « Transport - Bicing, » [En línia]. Disponible a: http://opendata.bcn.cat/opendata/es/catalog/ext/TRANSPORT/bicing/ [Últim accés: 18/06/2016]

[26] Zaragoza , « 2.3.3 Condiciones abiertos de Zaragoza, » [En línia]. Disponible a: http://www.zaragoza.es/ciudad/servicios/avisolegal.htm#condiciones [Últim accés: 18/06/2016]

[27] Joan Salvatella Ibáñez (08/05/2014). *FIB Master thesis: Design of a community sensor network*. Barcelona.

[28] Nancy Zeas Orellana (14/01/2015). *FIB Tesis de master: Desarrollo de una Plataforma Web para una Red de Monitorización de la Calidad del Aire*. Barcelona.

[29] David Wood, Marsha Zaidman, Like Ruth and Michael Hausenblas (2014). *Linked Data: Structured data on the Web*. USA, NY.

Índex d'imatges

Índex de taules